**UNIVERSITÀ POLITECNICA DELLE MARCHE**
**FACOLTÀ DI INGEGNERIA**
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione


**TESI DI LAUREA**

# Un framework per l'analisi della qualità dell'aria e per l'individuazione del percorso meno inquinato in una grande città

# A framework for analyzing air quality and identifying the least polluted route in a city

Relatore

Prof. Domenico Ursino

Candidato

Cristian Cingolani

**ANNO ACCADEMICO 2022-2023**

*Chi davvero vuole può. Ma per potere e volere,*
*bisogna prima sapere e, soprattutto, osare.*

Hernan Huarache Mamani, "La profezia della curandera"

**Sommario**

Nonostante le risorse impiegate negli ultimi anni, i livelli di smog nelle città italiane sono ancora troppo alti. Inoltre, si è molto lontani dai limiti normativi previsti da enti internazionali come l'Unione Europea. In particolare, la città di Torino, negli ultimi anni, è sempre stata sul podio delle città più inquinate d'Italia. Prendendo come riferimento tale città ed utilizzando dati storici sull'inquinamento e su fattori meteorologici, è stata eseguita una analisi per comprendere come questi dati siano correlati tra loro e per verificare la presenza di trend significativi. Una delle soluzioni proposte, da parte delle istituzioni, per combattere l'inquinamento, prevede l'impiego di mezzi pubblici e di veicoli ad impatto quasi nullo, come le biciclette. Nell'ottica di salvaguardare la salute dei cittadini, che decidono di adottare quest'ultima soluzione, si è voluto sviluppare un dispositivo hardware basato su Arduino che acquisisce i valori di PM10 dell'aria, la relativa posizione ed altre informazioni utili. Dopo aver superato una fase di elaborazione, questi dati vengono trasferiti ad un database e successivamente assegnati ai nodi di una mappa. Tale mappa rappresenta il quartiere San Salvario a Torino ed è stata creata su NetworkX. A partire dai dati storici delle stazioni fisse e da quelli di Arduino sono stati creati quattro servizi. Questi ultimi permettono agli utenti di ottenere informazioni sui valori di PM10 passati, presenti e futuri. L'utente può, inoltre, indicare un punto di partenza e di arrivo e verrà calcolato il percorso con livelli di inquinamento più bassi.

**Keyword**: PM10, Arduino, GPS, Data Analysis, Prediction, NetworkX, Pandas

# Indice

**Introduzione** **1**

**1 Inquinamento dell'aria nelle città** **3**
1.1 Situazione mondiale ed italiana . . . 3
1.2 Impatto sulla salute . . . 5
1.3 Evoluzione dei trasporti . . . 7

**2 Dati acquisiti da stazioni fisse** **9**
2.1 Raccolta dati . . . 9
2.2 Creazione del dataset . . . 11
2.3 Analisi dei dati . . . 14
2.3.1 Correlazione inquinamento – fattori meteorologici . . . 14
2.3.2 Serie temporali . . . 17

**3 Dati puntuali acquisiti da Arduino** **31**
3.1 Componenti hardware . . . 31
3.2 Codice implementativo . . . 36
3.2.1 Fase di setup . . . 37
3.2.2 Fase di loop . . . 37
3.3 Real Time Processing su NodeRed . . . 40
3.4 Raccolta dei dati . . . 41

**4 Creazione grafo della rete stradale** **43**
4.1 Grafo NetworkX . . . 43
4.2 Analisi del grafo . . . 44
4.3 Associazione valori stazioni fisse . . . 47
4.4 Associazione valori puntuali e costruzione dei dataset . . . 48

**5 Servizi per gli utenti** **52**
5.1 Premessa . . . 52
5.2 Richiesta dati PM10 per un particolare giorno . . . 53
5.3 Richiesta dati PM10 per un particolare giorno ed ora . . . 55
5.4 Richiesta percorso meno inquinato . . . 56
5.5 Richiesta heatmap sulla distribuzione di PM10 . . . 59

**6 Conclusioni e possibile evoluzione del progetto** **62**

**Bibliografia** **65**

**Ringraziamenti** **67**

# Elenco delle figure


1.1 Obiettivi sugli agenti inquinanti redatti dall'OMS . . . . . 4
1.2 Andamento delle emissioni di CO2 per Paese . . . . . 5
1.3 Stima degli anni di vita persi da ciascun cittadino a causa dell'inquinamento, per ciascun paese dell' UE . . . . . 6
1.4 Evoluzione delle emissioni di agenti inquinanti da parte dei trasporti . . . . . 7

2.1 Mappa sulla collocazione geografica delle stazioni fisse . . . . . 10
2.2 Grafico di correlazione inquinamento - fattori meteorologici . . . . . 15
2.3 Grafico tra il PM10 e la temperatura media giornaliera . . . . . 17
2.4 Grafico tra il PM10 ed i millimetri di pioggia giornalieri . . . . . 18
2.5 Grafico tra il PM10 e la velocità media del vento giornaliera . . . . . 19
2.6 Dataframe iniziale rappresentativo dei dati settimanali di PM10 e fattori meteorologici . . . . . 19
2.7 Dataframe con la data come indice . . . . . 20
2.8 Riassunto dei valori degli indici del dataframe . . . . . 20
2.9 Dati della prima riga del dataframe . . . . . 21
2.10 Dati dell'ultima riga del dataframe . . . . . 21
2.11 Informazioni statistiche sui valori dei campi del dataframe . . . . . 22
2.12 Valori del test ADF e p-value relativi ai dati di PM10 . . . . . 22
2.13 Visualizzazione andamento temporale PM10, vento, pioggia e temperatura . 23
2.14 Variazione dei valori di PM10 . . . . . 24
2.15 Grafico di autocorrelazione per il PM10 . . . . . 25
2.16 Grafico di autocorrelazione parziale per il PM10 . . . . . 26
2.17 Parametri calcolati da auto arima per creazione del modello ARIMA . . . . . 26
2.18 Analisi auto arima per la creazione del modello ARIMA . . . . . 27
2.19 P-value e sottocomponenti del modello ARIMA . . . . . 27
2.20 Visualizzazione e distribuzione dei residui . . . . . 28
2.21 Predizione del modello ARIMA confrontata con i valori reali di PM10 . . . . . 28
2.22 Visualizzazione media e deviazione standard . . . . . 29
2.23 Grafico di separazione fra trend, stagionalità e residui . . . . . 29
2.24 Visualizzazione degli indici del training set . . . . . 30
2.25 Grafico di predizione del modello SARIMA sul PM10 . . . . . 30

3.1 Specifiche del modulo Wi-Fi NINA-W102 . . . . . 33
3.2 Immagine di Arduino UNO WiFi Rev.2 . . . . . 34

3.3 Immagine del sensore PMS7003 per il rilevamento della PM10 . . . . . 34
3.4 Immagine di Adafruit Ultimate GPS Breakout . . . . . 35
3.5 Immagine del collegamento tra il GPS e l'antenna esterna . . . . . 36
3.6 Schema dei collegamenti tra le componenti . . . . . 37
3.7 Foto con focus su alcune componenti e collegamenti . . . . . 38
3.8 Foto del risultato finale in cui sono visibili tutti i collegamenti . . . . . 39
3.9 Foto della scatola chiusa con pannello in plexiglass . . . . . 40
3.10 Schema della fase di setup del codice Arduino . . . . . 41
3.11 Schema della fase di loop del codice Arduino . . . . . 41
3.12 Flussi di elaborazione real time in Node-RED . . . . . 42

4.1 Grafo creato su NetworkX per il quartiere San Salvario di Torino . . . . . 44
4.2 Numero di nodi ed archi che compongono il grafo . . . . . 45
4.3 Diametro del grafo . . . . . 45
4.4 Verifica della connessione del grafo . . . . . 45
4.5 Raggio, centro, chiusura delle triadi e periferia del grafo . . . . . 46
4.6 I dieci nodi con i valori più alti di centralità . . . . . 46
4.7 Cammino più corto tra i migliori tre nodi per centralità . . . . . 46
4.8 Coefficiente di clustering medio . . . . . 47
4.9 Distribuzione dei coefficienti di centralità dei nodi . . . . . 47
4.10 Mappa dei coefficienti di centralità dei nodi . . . . . 48

5.1 Menù di scelta servizi . . . . . 52
5.2 Esempio del servizio 1: data che non rispetta la data minima e assenza di dati in Arduino . . . . . 55
5.3 Esempio del servizio 1: data in cui sono presenti sia dati delle stazioni fisse che di Arduino . . . . . 55
5.4 Esempio del servizio 1: data troppo futura per eseguire predizioni . . . . . 56
5.5 Esempio del servizio 1: data in cui è possibile eseguire predizioni . . . . . 56
5.6 Esempio del servizio 2: data che non rispetta la data minima e assenza di dati Arduino . . . . . 57
5.7 Esempio del servizio 2: data in cui sono presenti sia dati delle stazioni fisse che di Arduino . . . . . 57
5.8 Esempio del servizio 2: data troppo futura per eseguire predizioni . . . . . 58
5.9 Esempio del servizio 2: data in cui è possibile eseguire predizioni . . . . . 58
5.10 Esempio di servizio 3: data in cui non sono presenti dati di Arduino . . . . . 58
5.11 Esempio di servizio 3: data in cui sono presenti dati di Arduino . . . . . 59
5.12 Esempio di servizio 3: mappa relativa ad una data in cui non sono presenti dati di Arduino . . . . . 59
5.13 Esempio di servizio 3: data futura in cui viene predetto il percorso a partire dai dati di Arduino . . . . . 60
5.14 Esempio di servizio 3: mappa relativa ad una data futura in cui viene predetto il percorso a partire dai dati di Arduino . . . . . 60
5.15 Esempio di servizio 4: data in cui sono presenti dati Arduino . . . . . 60
5.16 Esempio di servizio 4: heatmap PM10 . . . . . 61

6.1 Mappa ESA sulla distribuzione di inquinamento in Europa . . . . . 63

# Elenco delle tabelle


2.1 Esempio di dati relativi al PM10 della stazione fissa Torino-Consolata dell'anno 2017 . . . . . 10
2.2 Esempio dei dati relativi alla quantità di pioggia giornaliera . . . . . 11
2.3 Esempio dei dati relativi alla temperatura giornaliera . . . . . 11
2.4 Esempio dei dati relativi al vento giornaliero . . . . . 11
2.5 Esempio di dati relativi al dataset finale delle stazioni fisse . . . . . 14
2.6 Esempio di dati relativi al dataset finale delle stazioni fisse . . . . . 20

3.1 Dati inviati dal sensore GPS . . . . . 39
3.2 Struttura del database CouchDB . . . . . 42

4.1 Esempio di dati relativi al dataset Arduino per il nodo 0 . . . . . 51

# Introduzione

L'esposizione alle particelle sottili da parte dei cittadini ha causato, negli anni, molti morti e l'incremento di alcune malattie respiratorie. In particolar modo, la situazione è critica nelle grandi città dove il traffico e le attività industriali sono maggiori. Questo problema riguarda tutto il mondo, ma la misura in cui ciascun Paese è vulnerabile ad esso è influenzata fortemente dalla situazione economica dello stesso. Ciò viene spiegato sia dalla possibilità di avere o meno un sistema sanitario efficiente, che riesce a rispondere prontamente all'incremento di alcune malattie, che dalla disponibilità economica del Paese di investire in nuove tecnologie che permettono di limitare, almeno localmente, le emissioni di agenti inquinanti.

Se si analizza l'Italia, si arriva alla conclusione che il problema è accentuato nelle grandi città del nord, come Torino e Milano. Esse sono, rispettivamente, prima e seconda città italiana per numero di giorni di sforamenti di PM10 in un anno. Sicuramente, la loro collocazione geografica favorisce il ristagno di polveri sottili, ma, non per questo, non è possibile intervenire ed arginare il fenomeno.

La soluzione più scontata, ma non di più semplice implementazione, è diminuire il traffico. Ciò è realizzabile a partire dall'impiego di mezzi pubblici e dall'incentivo, da parte delle istituzioni, dell'impiego di biciclette, o di mezzi analoghi, ad emissioni zero. Per rendere possibile quest'ultima soluzione, bisogna convincere i cittadini a preferire tale modalità di spostamento, con incentivi e con la costruzione di piste ciclabili.

Attualmente, la maggior parte delle persone non adottano questa soluzione, ma, per chi lo fa, può essere messo a disposizione uno strumento che indichi quale sia il tragitto migliore dal punto di vista salutare, o quali siano le aree con livelli di inquinamento più alti. Tale tragitto, probabilmente, è anche il percorso con meno traffico. Il tutto deve tenere in considerazione anche l'orario; molto spesso alcune strade si congestionano solo in particolare fasce di orario, che spesso corrispondono allo spostamento da casa verso il luogo di lavoro, e viceversa.

Questa tesi mira, inizialmente, ad analizzare il trend del PM10 e la correlazione tra esso e i fattori come la stagione, la pioggia, il vento e la temperatura. Ciò è particolarmente interessante perchè permette di capire l'andamento futuro dei valori di inquinamento e, quindi, di monitorare l'efficacia delle politiche adottate nel tempo. Inoltre, comprendere come il PM10 vari in funzione della stagione, e conseguentemente dei vari fattori meteorologici, consente di migliorare i modelli predittivi.

In seguito a questa attività di analisi, si pone l'attenzione sulla costruzione di un dispositivo basato su Arduino. Tale dispositivo sfrutta un sensore PM10 ed un sensore GPS, per acquisire dati di PM10 ed associare ciascun dato ad una posizione. I dati vengono trasmessi, tramite una connessione Wi-Fi, a Node-RED, che provvede all'elaborazione ed al salvataggio

degli stessi in un database CoutchDB. Successivamente, ciascun dato di PM10, grazie alla posizione di acquisizione, viene associato ad un nodo della mappa della città di interesse. Nel caso specifico si usa una mappa in NetoworkX, del quartiere San Salvario a Torino, in cui ciascun nodo rappresenta un incrocio, o il termine di una strada chiusa, e ciascun arco rappresenta una strada. A ciascun nodo vengono assegnati anche i dati delle stazioni fisse; questo per tenere traccia dello storico e rendere ciascuno di essi indipendente nel recupero dei dati.

A partire da ciò, nel primo servizio, l'utente può indicare il nodo e la data di interesse e ricevere il valore di PM10. La data indicata può essere passata, presente o futura; in quest'ultimo caso viene sfruttato un modello predittivo ARIMA.

Un secondo servizio ha un funzionamento analogo al primo, ma l'utente può specificare l'ora di interesse.

Il terzo servizio sviluppato permette all'utente di indicare il nodo di partenza, il nodo di arrivo e la data di interesse. Dunque, si restituisce il percorso con i livelli di PM10 più bassi. Questo è particolarmente importante per il perseguimento dell'obiettivo di salvaguardare la salute dei cittadini che decidono di spostarsi con mezzi a basso impatto ambientale, come la bici.

Viene, infine, creato un quarto servizio che permette di visualizzare una heatmap sulla distribuzione dei valori di PM10. Questo è utile anche alle amministrazioni per individuare criticità ed intervenire gestendo in modo diverso la viabilità.

La presente tesi è composta da sei capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 sarà analizzato il problema dell'inquinamento nelle grandi città; verranno indagati l'evoluzione temporale degli agenti inquinanti nella varie aree del mondo, le tematiche relative ai problemi di salute che l'inquinamento causa e le innovazioni che sono state introdotte nel campo dei trasporti per contrastare il problema.

- Nel Capitolo 2 verranno analizzati i dati relativi all'inquinamento ed a fattori meteorologici, provenienti da stazioni fisse. A partire da tali dati verranno messi in evidenza l'andamento temporale, le correlazioni presenti tra essi e si proverà a costruire un modello predittivo basato sui valori di PM10.

- Nel Capitolo 3 verranno percorsi i passi che hanno portato alla costruzione di un dispositivo basato su Arduino. Tale dispositivo sarà utile per ottenere e salvare i dati puntuali sui livelli di PM10 e sulla posizione di acquisizione degli stessi.

- Nel Capitolo 4 verrà creato un grafo NetworkX di un quartiere della città di Torino e, a ciascun nodo, verranno associati i valori di inquinamento acquisiti sia dalle stazioni fisse che da Arduino.

- Nel Capitolo 5 verranno descritti i servizi creati per gli utenti, effettuando di volta in volta alcuni test. Tali servizi permettono di ottenere i dati di PM10 di un nodo in una determinata data, il percorso con valori di inquinamento più bassi ed una heatmap sulla distribuzione della PM10.

- Nel Capitolo 6 verranno tratte le conclusioni e verranno delineati alcuni possibili sviluppi futuri.

CAPITOLO 1

# Inquinamento dell'aria nelle città

*In questo capitolo iniziale si introdurrà il problema dell'inquinamento nelle città. Le città sono i luoghi in cui le attività industriali sono più sviluppate ed il numero di veicoli che transita è elevato. Conseguentemente, i livelli degli agenti inquinanti, come il particolato, sono molto alti. Si vedrà come diversi enti internazionali e nazionali hanno imposto dei limiti per arginare il problema e come, conseguentemente, si sono evoluti i valori di inquinamento nel mondo ed in Italia.*

*Questi agenti hanno un impatto sulla salute dell'uomo, e quindi, verranno analizzati i principali rischi che derivano da una esposizione elevata ad essi. Verranno mostrati gli incrementi della probabilità di contrarre alcune malattie ed il numero di anni persi a causa dell'inquinamento, per ciascun paese dell'UE.*

*Il miglioramento di tale situazione passa sia dall'evoluzione tecnologica nel mondo dei trasporti che dal cambiamento delle abitudini dei cittadini. Verranno, dunque, messi a confronto diversi metodi di spostamento comparando il numero di grammi di anidride carbonica emessi per chilometro e per passeggero.*

*In quest'ottica, verrà messo in luce come la bici, e soluzioni simili, siano molto spesso da preferire per spostamenti medio-corti. Infine, verrà esaminata l'evoluzione dei trasporti pubblici in alcune città in quanto costituiscono un tassello fondamentale nel miglioramento della qualità dell'aria.*

## 1.1 Situazione mondiale ed italiana

Il problema dell'inquinamento delle città è un tema globale e, per questa ragione, l'OMS nel 2022 ha pubblicato delle linee guida sulla qualità dell'aria, con lo scopo di offrire raccomandazioni quantitative basate su considerazioni di carattere sanitario. Gli inquinanti presi in esame sono: particolato (PM2,5 e PM10), ozono, biossido di azoto, anidride solforosa e monossido di carbonio.

Sempre l'OMS mette in evidenza come nei paesi più ricchi i livelli di inquinamento si sono abbassati nel tempo; nei paesi più poveri ed in via di sviluppo è avvenuto spesso il contrario. Da questo deriva la necessità di imporre dei limiti che siano molto più severi degli standard nazionali in vigore in molte parti del mondo. Nonostante in alcune aree del mondo siano stati osservati miglioramenti nella qualità dell'aria, il numero di vittime e di anni di vita in salute persi è rimasto stabile dagli anni '90 a oggi.

La Figura 1.1 mostra i livelli raccomandati di alcuni inquinanti e gli obiettivi intermedi.

È interessante comparare i valori limite suggeriti dall'OMS con i valori raggiunti dagli stati con i livelli di PM2.5 più alti nel 2022. Nei dati pubblicati da IQAir troviamo il Chad in prima posizione con 89.7 $\mu g/m^3$, seguito dall' Iraq con 80.1 $\mu g/m^3$ e dal Pakistan con 70.9 $\mu g/m^3$. I paesi più popolosi al mondo, cioè India e Cina, si posizionano, rispettivamente, in ottava e venticinquesima posizione con valori di PM2.5 pari a 53.3 $\mu g/m^3$ ed a 30.6 $\mu g/m^3$. Scorrendo la classifica si osserva come nelle prime posizioni vi siano principalmente Paesi

| Inquinante | Tempo di media | Obiettivo intermedio | | | | Livello AQG |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | |
| **$PM_{2,5}$, µg/m³** | Annuale | 35 | 25 | 15 | 10 | 5 |
| | 24 ore[a] | 75 | 50 | 37,5 | 25 | 15 |
| **$PM_{10}$, µg/m³** | Annuale | 70 | 50 | 30 | 20 | 15 |
| | 24 ore[a] | 150 | 100 | 75 | 50 | 45 |
| **$O_3$, µg/m³** | Picco stagionale[b] | 100 | 70 | – | – | 60 |
| | 8 ore[a] | 160 | 120 | – | – | 100 |
| **$NO_2$, µg/m³** | Annuale | 40 | 30 | 20 | – | 10 |
| | 24 ore[a] | 120 | 50 | – | – | 25 |
| **$SO_2$, µg/m³** | 24 ore[a] | 125 | 50 | – | – | 40 |
| **CO, mg/m³** | 24 ore[a] | 7 | – | – | – | 4 |

[a] 99° percentile (ovvero 3-4 giorni di superamento all'anno).

[b] Media della concentrazione media giornaliera massima su 8 ore di $O_3$ nei sei mesi consecutivi con la più alta concentrazione media mobile semestrale di $O_3$.

**Figura 1.1:** Obiettivi sugli agenti inquinanti redatti dall'OMS

in via di sviluppo, come anticipato in precedenza. L'Italia si posiziona in cinquantaduesima posizione con 18.9 µg/m³, mentre gli USA in novantanovesima posizione con 8.9 µg/m³.

Quest'ultimo dato è molto interessante perché, come vedremo, gli USA sono tra gli stati che emettono più sostanze inquinanti nell'aria, ma, nonostante ciò, la qualità dell'aria non è delle peggiori (ovviamente con eccezioni se si parla delle grandi metropoli americane). In un articolo pubblicato da Selectra viene riportato come i paesi che immettono nell'atmosfera più quantitativi di CO2 sono Cina, USA, India, Russia e Giappone. Impressionante è il valore raggiunto dalla Cina ovvero 9.9 miliardi di tonnellate di CO2 riversate nell'atmosfera su un totale globale di circa 32 miliardi di tonnellate. Se, però, si guarda il valore per capita abbiamo al vertice il Qatar, seguito da Kuwait, Arabia Saudita, Canada, USA, Germania, Cina, Spagna e Francia.

Una visione globale sull'andamento delle emissioni di CO2 nel tempo si ha nel grafico in Figura 1.2, elaborato da Ourworldindata.

Appare evidente come, soprattutto a partire dagli anni 50 del ventesimo secolo, sia avvenuta una impennata delle emissioni. Questo non è giustificato solo dall'incremento della popolazione mondiale, da 2.6 miliardi del 1950 ai circa 8 miliardi attuali, ma, soprattutto, dall'incremento dell'attività industriale e del traffico. Vedendo l'andamento dell'Unione Europea si può notare come, in seguito all'incremento avvenuto negli anni del boom economico, le emissioni siano rimaste piuttosto costanti con una inversione di trend avvenuta negli ultimi decenni. Quello visto fino a questo momento è un quadro globale che permette di comprendere l'evoluzione nel tempo e nello spazio delle emissioni di particolato, anidride carbonica, etc., ma è importante approfondire la situazione presente nelle grandi città del nord Italia, in particolare Torino. Il focus su queste aree è dovuto al livello molto alto di alcuni inquinanti causato dalla combinazione di una intensa attività industriale e di una particolare morfologia del territorio. La pianura padana, infatti, è situata tra la catena montuosa delle

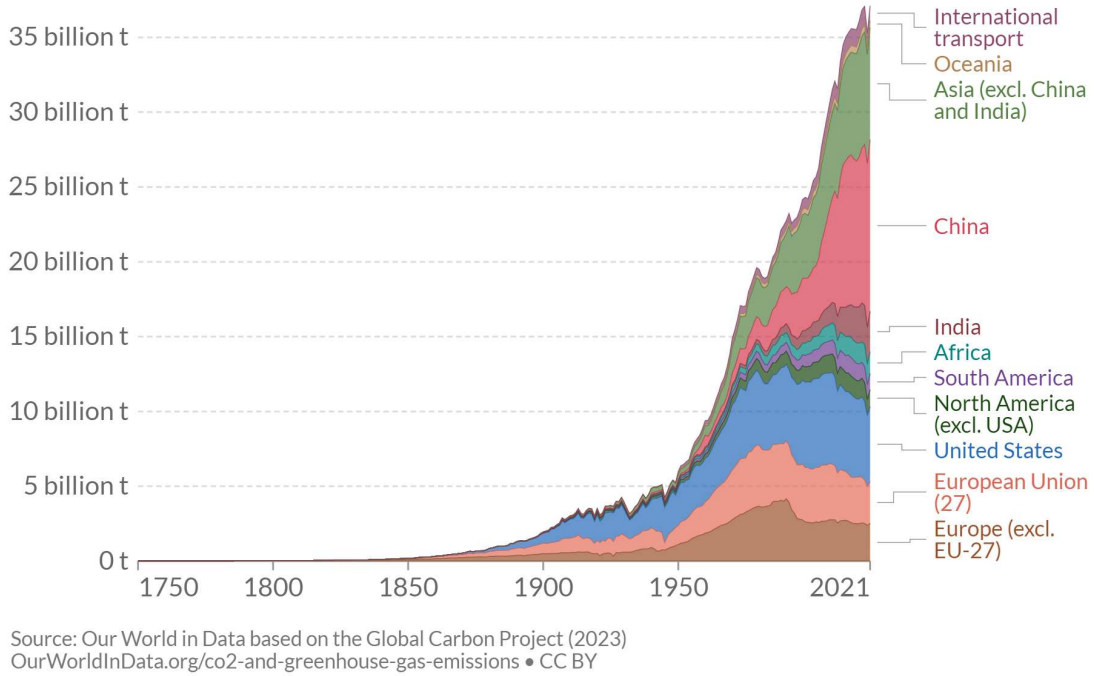


**Figura 1.2:** Andamento delle emissioni di CO2 per Paese

Alpi e quella degli Appennini, non permettendo, così, ai venti di "pulire" l'aria e portare via gli agenti inquinanti. Non sorprende che le situazioni peggiori si trovino a Torino, Milano, Modena, Asti, Padova e Venezia, le quali hanno registrato più del doppio degli sforamenti consentiti. In particolare, Torino si piazza al primo posto con 98 giorni di sforamento, seguita da Milano con 84, Asti 79, Modena 75, Padova e Venezia con 70.

Inoltre, rispetto ai nuovi target europei del 2030, sarebbero fuorilegge il 76% delle città per il PM10, l'84% per il PM2.5 e il 61% per l'NO2. Per non superare i target europei e dell'OMS (20 µg/mc per il PM10, 10 µg/mc per il PM2.5, 20 µg/mc per l'NO2) Torino e Milano dovrebbero ridurre i valori complessivi attuali di circa il 43%, numero molto difficile da raggiungere. Legambiente stima che ci vorrebbero altri 17 anni, in media, per raggiungere tali obiettivi.

## 1.2 Impatto sulla salute

Uno degli scopi del progetto è salvaguardare la vita dei cittadini e, per capirne l'importanza, è necessario analizzare gli impatti dell'inquinamento sulla salute umana.

Numerosi studi, infatti, dimostrano come l'aumento delle malattie cardiovascolari e respiratorie sia correlato con una qualità dell'aria scarsa. In particolare, ictus, infarti, ipertensione, scompenso cardiaco, fibrillazione atriale e tromboembolismo venoso sono solo alcuni degli effetti negativi.

L'OMS riporta come un quarto dei decessi per ictus e per malattie cardiache ha correlazione con l'inquinamento. Nel 2013 l'Agenzia per la ricerca sul cancro dell'OMS, lo IARC, ha inserito l'inquinamento e il particolato tra i cancerogeni certi per l'uomo. Studi su centinaia di migliaia di persone durati anni hanno dimostrato come il 29% dei decessi per tumore al polmone siano causati proprio dall'esposizione a PM10 ed a PM2.5.

Il "Country profiles of the environmental burden of disease" è il primo rapporto sull'impatto delle condizioni ambientali sulla salute ed analizza i dati Paese per Paese. E' stato redatto dall'OMS nel 2007 ed i dati mostrano come i paesi più colpiti siano quelli a basso

reddito, come Angola, Burkina Faso, Mali e Afghanistan. Si stima che in alcuni Paesi un miglioramento ambientale, quindi anche dell'inquinamento, potrebbe prevenire circa un terzo delle malattie.

Anche la Corte dei conti Europea si è interessata al tema ed in una relazione esordisce dicendo: "L'inquinamento atmosferico costituisce il principale rischio ambientale per la salute nell'Unione europea (UE) ". Sicuramente la frase appena citata è piuttosto forte e provocatoria ma, se si analizzano i dati più a fondo, si scopre come la realtà sia proprio questa. Infatti, questa problematica determina in UE circa 400000 decessi prematuri, dunque più di 1000 al giorno, e comporta diseconomie legate alla salute per centinaia di miliardi di euro, per la precisone tra i 330 ed i 940 miliardi di euro all'anno. Nel 2014 si stima che la maggior parte dei decessi prematuri siano attribuibili al PM2,5, all'NO2 e all'O3. Nel testo viene ribadito un concetto cardine di questo progetto, ovvero la forte esposizione delle zone urbane a tali rischi.

Infine, in Figura 1.3 vengono mostrati gli anni di vita, in buona salute, persi a causa dell'inquinamento dell'aria.

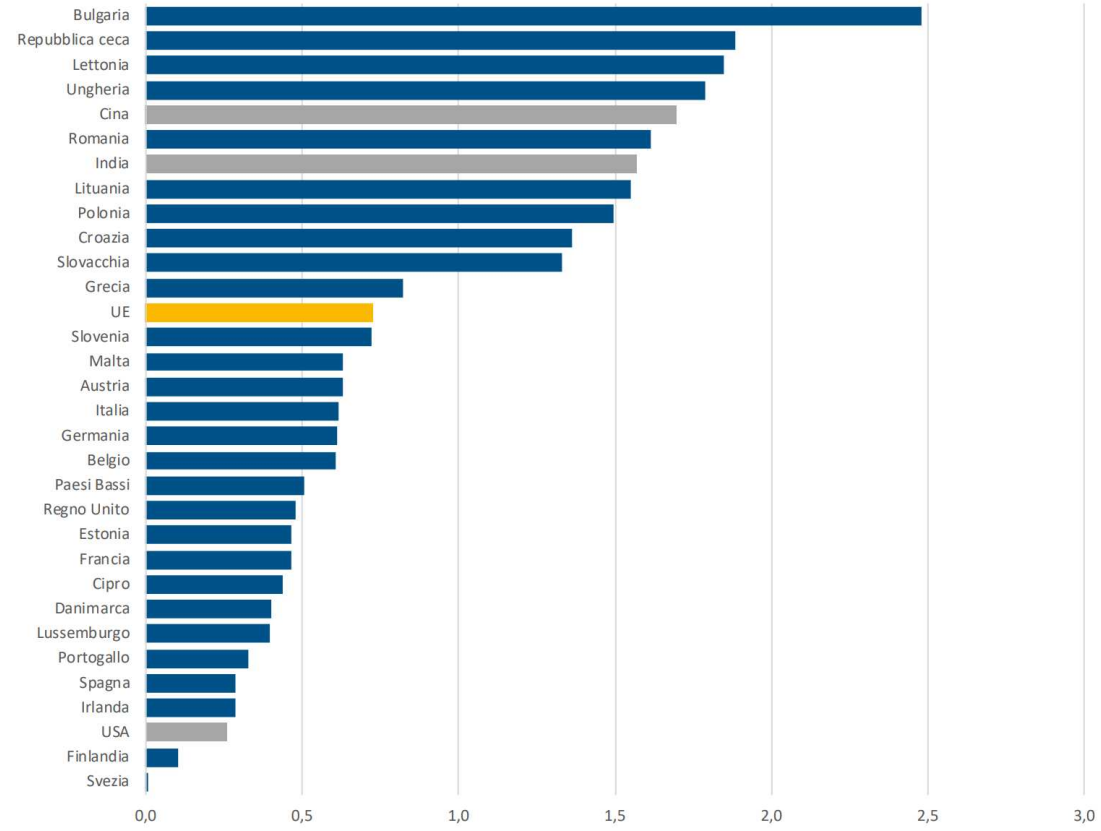


**Figura 1.3:** Stima degli anni di vita persi da ciascun cittadino a causa dell'inquinamento, per ciascun paese dell' UE

Al primo posto abbiamo la Bulgaria che stacca di gran lunga gli altri Paesi con un valore di quasi 2.5 anni persi. L'Italia si posiziona quasi a metà classifica con più di 0.6 anni persi. La Svezia si conferma, invece, il Paese meno interessato dal fenomeno. Anche gli USA vengono inseriti nel grafico, come metro di paragone, ed, ancora una volta, viene confermato come siano meno impattati, dal punto di vista salutare, di molti altri Paesi.

Per concludere questa sezione è doveroso ricordare come gli impatti sulla salute dell'uomo non siano solo diretti, ma anche indiretti. Fenomeni come le piogge acide possono provocare, al contatto con la pelle, irritazioni e possono danneggiare gravemente l'ambien-

te, vegetazione compresa. Ancora una volta questi eventi sono dovuti principalmente al consumo di combustibili fossili, al funzionamento di impianti industriali, agli incendi che scoppiano nelle foreste e nelle savane e all'attività di centrali elettriche e autoveicoli.

## 1.3 Evoluzione dei trasporti

Prima di addentrarci nello sviluppo del progetto torniamo ad uno degli argomenti core dell'elaborato ovvero l'inquinamento causato dal trasporto.

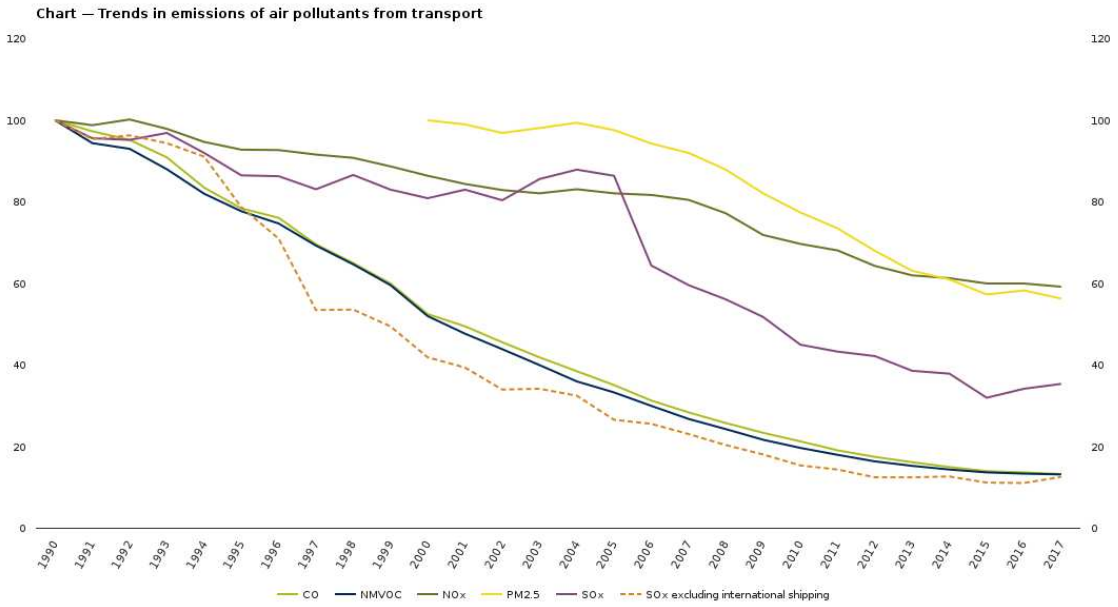


**Figura 1.4:** Evoluzione delle emissioni di agenti inquinanti da parte dei trasporti

Dalla Figura 1.4, ottenuta dal portale dell'Agenzia Europea dell'Ambiente, si può facilmente evincere come l'evoluzione nel settore dei trasporti abbia portato ad una diminuzione di praticamente tutte le sostanze inquinanti emesse. Questo miglioramento tecnologico porta subito alla mente l'introduzione di motori elettrici, i quali hanno avuto successo soprattutto negli ultimi anni. Però questo aspetto non può aver influito in maniera importante sull'andamento decrescente dei decenni precedenti, dove l'introduzione di nuovi motori termici e di filtri antiparticolato sempre più efficienti hanno giocato un ruolo chiave.

Questo miglioramento non è passato solo dal trasporto privato, ma anche dal trasporto pubblico. Soluzioni efficienti ed in grado di trasportare una grande quantità di persone sono ancora oggi al centro del dibattito sullo sviluppo di città sempre più interconnesse e green. Basti pensare al periodico rafforzamento ed ammodernamento del trasporto tramite bus e tram. Prendendo come esempio la città di Torino, è stato presentato il piano del "Nuovo Trasporto per Torino" da parte di GTT (società che si occupa del servizio di trasporto pubblico locale nel territorio comunale e nell'area metropolitana torinese). Questo piano prevede lo stanziamento di oltre un miliardo di euro per prolungare la linea 1 della metropolitana, per 2 nuove linee tranviarie e 30km di binari in più e per 437 nuovi bus ecologici. Menzione d'onore va alla metropolitana in quanto possiamo dire con certezza che l'introduzione di questo mezzo di trasporto, avvenuto il 10 gennaio 1863 a Londra, ha cambiato il volto di moltissime città. Basti pensare alla rapidità con cui Milano è riuscita a sviluppare la sua rete negli ultimi anni, arrivando ad avere 5 linee e ben oltre 100km di estensione. Torino, invece, ha approvato la variante urbanistica per dare il via alla costruzione della linea 2.

Una domanda che potrebbe sorgere spontanea riguarda i quantitativi di CO2 emessi per ciascuna modalità di spostamento. Secondo alcune indagini la bici ha una emissione praticamente pari a zero e la e-bike mantiene anch'essa valori bassi di inquinamento. Esclusa la bici, il mezzo più green è, senza dubbio, il treno; infatti, un treno nazionale produce 40 g/km di CO2 a persona, mentre la metro solo 20 g/km di CO2. Treni più moderni, come i Frecciarossa, emettono solo 28 g/km di CO2. L'autobus urbano inquina parecchio, 104 g/km di CO2 per passeggero, mentre il filobus fa meglio e si attesta a 46 g/km di CO2. Gli autobus di linea, così come i Flixbus ed altri autobus per i viaggi a lunga percorrenza, emettono solo 27 g/km CO2 per passeggero. Un'auto di media taglia emette 168 grammi di CO2 per ogni chilometro, valore piuttosto alto e tra le principali cause di congestioni del traffico e di inquinamento nelle città. Condividere la macchina con altre persone può rendere questo mezzo di trasporto più eco-friendly. Non è oggetto di interesse di questo elaborato, ma è doveroso citare i livelli di inquinamento degli aerei. Si passa da 254 g/km di CO2, per le tratte sotto le 2 ore, a 195 g/km di CO2, per quelle più lunghe.

Nelle grandi città spesso si concentra il traffico nelle ore di entrata ed uscita dai luoghi di lavoro creando congestioni ed incrementando le emissioni totali. Uno studio della Nation Association of City Transportation ha contato il numero di persone che si possono spostare in una determinata strada per ciascuna modalità di trasporto, non tenendo, però, conto della velocità del traffico. Se in una corsia di veicoli privati si potrebbero spostare da 600 a 1600 persone all'ora, in una corsia di autobus si potrebbero muovere fino a 8000 persone all'ora, fino a 7500 persone in bici, fino a 9000 persone a piedi e fino a 25000 persone con i mezzi su rotaia. Questi dati mostrano come soluzioni alternative all'auto permettono di efficientare il trasporto pubblico ed eventualmente eliminare delle corsie per far posto ad aree verdi o percorsi pedonali/ciclabili.

Prima di concludere questa introduzione sull'inquinamento nelle città è necessario comprendere la situazione italiana sullo sviluppo delle piste ciclabili. Ferrara si aggiudica il primo posto nella classifica delle città con più piste ciclabili per abitante con un rapporto di 1,14 m/ab. Al secondo posto troviamo Reggio Emilia con 1,13 m/ab ed al terzo Modena con 1.07 m/ab. Torino, che sarà la città presa in esame in questo elaborato, rende possibile muoversi in bicicletta grazie a 258 km di piste ciclabili e ciclo-pedonali. Questo aspetto è rilevante perché rientra tra le ragioni per cui i cittadini potrebbero preferire la bici ad altre soluzioni.

Come è stato ampiamente spiegato, incentivare l'uso della bicicletta come mezzo di trasporto quotidiano significa diminuire progressivamente gli spostamenti motorizzati ed avere città più pulite, silenziose, sicure e vivibili. Sulle brevi e medie distanze, fino ai 6-7 km, la bici è competitiva rispetto ai mezzi motorizzati. Si ricorda a tal proposito che la maggior parte degli spostamenti giornalieri non supera tale distanza. Per concludere, l'uso quotidiano della bicicletta migliora la salute di chi la usa, riducendo il rischio di malattie cardiache e consentendo di respirare meno inquinanti.

# CAPITOLO 2

## Dati acquisiti da stazioni fisse

*Lo scopo di questo capitolo è recuperare dati storici sull'inquinamento e su fattori meteorologici per poter fare una analisi e creare un modello SARIMA in grado di eseguire una predizione sui livelli di inquinamento futuri nella città di Torino. Tale predizione riguarderà esclusivamente il PM10, ovvero l'insieme delle sostanze, solide o liquide, sospese in aria e che hanno dimensione di alcune decine di µm. La ragione di tale scelta, rispetto al PM2.5, risiede nella presenza di più dati a disposizione. Ricordiamo come le particelle PM10, 10–100 µm, penetrano nel tratto superiore delle vie aeree, mentre le particelle PM2,5, <4 µm, possono giungere fino alle parti inferiori dell'apparato respiratorio.*

*I dati sull'inquinamento, che sono stati utilizzati per l'analisi, sono resi pubblici nel portale della regione Piemonte denominato Aria e sono stati selezionati i dati di tre stazioni fisse nel territorio di Torino, per poterne verificare la correttezza. I dati su pioggia, vento e temperatura sono stati richiesti direttamente via mail ad Arpa Piemonte. In entrambi i casi è stato necessario un lavoro di pulizia ed interpolazione dei dati, nell'ottica di ottenere dati di qualità e, di conseguenza, poter eseguire una analisi corretta e ricavare un modello efficace.*

*Il motivo per cui questo capitolo gioca un ruolo fondamentale nel progetto è che non è stato possibile avere uno storico di molti mesi, o addirittura anni, dei dati puntuali provenienti da Arduino. Dunque, solo grazie a dati provenienti da stazioni fisse si può comprendere la correlazione tra inquinamento ed i fattori meteorologici, nonché il trend di medio-lungo periodo (anni).*

## 2.1 Raccolta dati

Come anticipato nell'introduzione del capitolo, i dati di PM10 sono stati scaricati dal sito Aria della regione Piemonte per le tre stazioni di interesse: Torino Lingotto, Torino Consolata e Torino Rubino.

In Figura 2.1 è visibile la mappa con la collocazione geografica delle tre stazioni fisse.

La scelta di queste stazioni è legata all'area presa in esame nell'elaborato, ovvero il quartiere San Salvario ed una piccola parte di Nizza Millefonti. Vediamo nel dettaglio che dati è in grado di raccogliere ciascuna stazione, ricordando che questa è una "foto" della situazione attuale; quindi, per alcuni parametri, i dati storici potrebbero essere limitati:

- TO- Consolata: Stazione di traffico in grado di misurare NOx, CO, SO2, PM10, (As-Cd-Ni-Pb), B(a)P, BTX, PTS
- TO-Rubino: Stazione di fondo in grado di misurare NOx, O3, CO, PM10, PM10ß, PM2,5ß, (As-Cd-Ni-Pb), B(a)P, BTX
- TO-Lingotto: Stazione di fondo in grado di misurare NOx, O3, PM10-PM10ß, PM2,5, (As-Cd-Ni-Pb), B(a)P, BTX

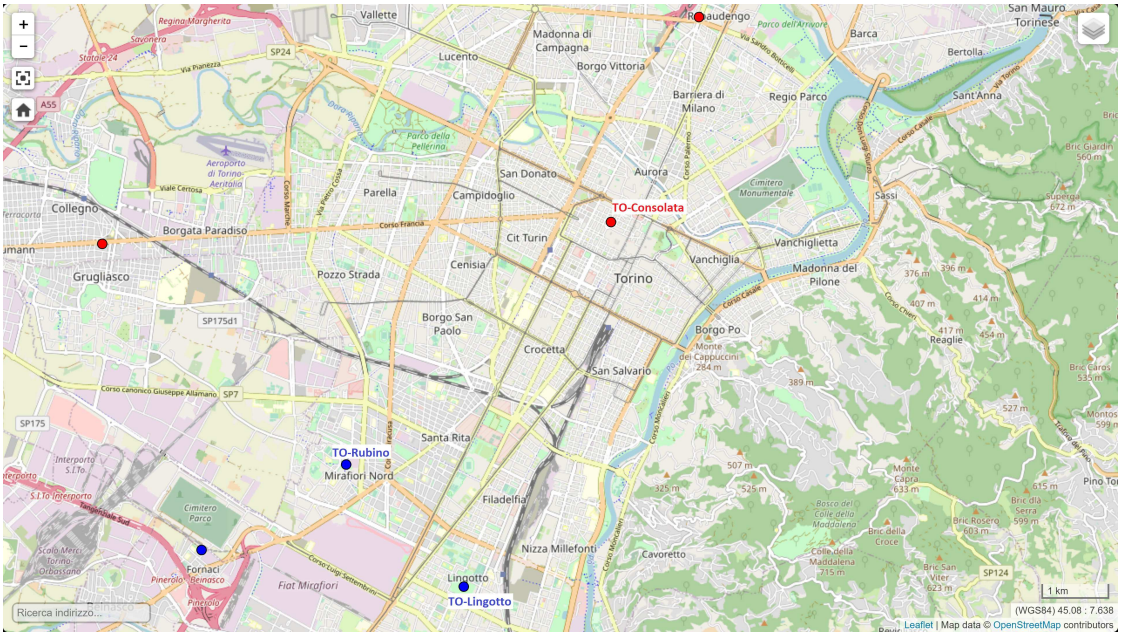


**Figura 2.1:** Mappa sulla collocazione geografica delle stazioni fisse

Il primo passo è scaricare questi dati, ma è possibile selezionare solo una stazione alla volta e per un periodo massimo di un anno. Quindi sono stati raccolti tutti i file `.csv` che rappresentano i dati a partire dal 1° gennaio 2017 e presentano una nomenclatura del tipo "<stazione-fissa>_Polveri-sottili_<data-inizio>_<data-fine>" (es: "Torino-Consolata_Polveri-sottili_2017-01-01_2017-12-31"). Ciascun file contiene i campi presenti nella Tabella 2.1.

| **Data rilevamento** | **Ora** | **Id Rete Monitoraggio** |
|---|---|---|
| 01/01/2017 | 23:59 | 13 |
| **Codice Istat Comune** | **Progr. Punto Comune** | **Denominazione Stazione** |
| 1272 | 803 | Torino - Consolata |
| **Id Parametro** | **Descr. Parametro** | **Id Un. misura** |
| PM10_GBV | PM10 - Basso Volume | 23 |
| **Descr. Un. misura** | **Valore** | **Stato** |
| microgrammi / metro cubo | 89.0 | V |

**Tabella 2.1:** Esempio di dati relativi al PM10 della stazione fissa Torino-Consolata dell'anno 2017

Solo alcuni campi come "Data rilevamento" o "Valore" verranno presi in esame, mentre altri, come "Id Parametro", verranno utilizzati per filtrare i dati di interesse.

Ottenuti i dati del PM10 ci si è concentrati su quelli relativi a fattori meteorologici, come pioggia, vento e temperatura. In questo caso i dati sono stati richiesti direttamente ad Arpa Piemonte sotto forma di file `.csv`, sempre con data a partire dal 1° gennaio 2017. In particolare, sono stati ottenuti tre file: richiesta_dati_gg_<pioggia/vento/temperatura>. Purtroppo, questi file non hanno una struttura che ne permette un immediato uso. In ciascuno sono presenti delle righe non strutturate e con informazioni superflue che dovranno essere eliminate in una fase successiva. Escluse le righe superflue, il file "richiesta_dati_gg_pioggia" si presenta secondo la struttura della Tabella 2.2.

| data | totale (mm) | classe |
|---|---|---|
| 01/01/2017 | 0.0 | Z |

**Tabella 2.2:** Esempio dei dati relativi alla quantità di pioggia giornaliera

A seguire, il file "richiesta_dati_gg_temperatura" si presenta secondo la struttura della Tabella 2.3.

| data | ora minimo (UTC) | Tmin (gradi C) |
|---|---|---|
| 01/01/2017 | 04:40 | -1.8 |
| **classe minimo** | **Tmed** | **classe media** |
| 0 | 1.9 | Z |
| **ora massimo (UTC)** | **Tmax** | **classe massimo** |
| 13:45 | 9.9 | 0 |

**Tabella 2.3:** Esempio dei dati relativi alla temperatura giornaliera

Infine, il file "richiesta_dati_gg_vento" si presenta secondo la struttura della Tabella 2.4.

| data | ora raffica (UTC) | vel. raffica (m/s) | direzione raffica (gradi) |
|---|---|---|---|
| 01/01/2017 | 19:14 | 3.3 | 13:45 |
| **classe raffica** | **vel. media (m/s)** | **classe media** | |
| 0 | 0.7 | Z | |

**Tabella 2.4:** Esempio dei dati relativi al vento giornaliero

Per tutti i file visti fino ad ora è necessaria una pulizia e una modifica dei dati ed una ristrutturazione globale che ha come fine la creazione di un dataset unico che permetta di procedere con le fasi successive nel modo più agile possibile.

## 2.2 Creazione del dataset

Seguendo il processo di ETL (Extract, Transform, Load), una volta estratti i dati è necessario trasformare questi ultimi per potere creare un dataset di qualità. Lo scopo di questo capitolo sarà proprio la creazione di un unico file `.csv` contenente sia dati giornalieri di PM10 che dati relativi ai fattori meteorologici.

La prima operazione che è stata svolta è il salvataggio di alcuni campi di ciascun file `.csv` relativo al PM10 nei corrispettivi array. In particolare, ciascun elemento dell'array è costituito, a sua volta, da un array composto da due elementi:

- campo in posizione zero, ovvero la data di rilevamento ;
- campo in posizione dieci, ovvero il valore di PM10.

Prima di salvare ciascuna coppia sono stati creati dei controlli che filtrino, ad esempio, solo le righe del file con campo "Id Parametro" pari a PM10_GBV. Questo perché ciascun file

può avere all'interno la stessa data ripetuta con i rispettivi valori di particolato ma calcolati a partire da parametri diversi.

A questo punto avremo i seguenti array:

- *ToConsolata2017Arr, ToConsolata2018Arr, ...*
- *ToLingotto2017Arr, ToLingotto2018Arr, ...*
- *ToRubino2017Arr, ToRubino2018Arr, ...*

Per verificare l'efficacia di questa selezione e pulizia dei dati, vengono stampate le dimensioni di ciascun array. Il risultato ottenuto nel terminale è il seguente:

- *2017*

  *ToConsolata2017Arr: 365 / ToLingotto2017Arr: 365 / ToRubino2017Arr: 365*
- *2018*

  *ToConsolata2018Arr: 365 / ToLingotto2018Arr: 365 / ToRubino2018Arr: 365*
- *2019*

  *ToConsolata2019Arr: 365 / ToLingotto2019Arr: 365 / ToRubino2019Arr: 365*
- *2020*

  *ToConsolata2020Arr: 366 / ToLingotto2020Arr: 366 / ToRubino2020Arr: 366*
- *2021*

  *ToConsolata2021Arr: 365 / ToLingotto2021Arr: 365 / ToRubino2021Arr: 365*
- *2022*

  *ToConsolata2022Arr: 365 / ToLingotto2022Arr: 365 / ToRubino2022Arr: 365*
- *2023*

  *ToConsolata2023Arr: 162 / ToLingotto2023Arr: 162 / ToRubino2023Arr: 162*

Le dimensioni degli array sono coerenti con il numero di giorni di ciascun anno. Questo permette di accorpare tutti i dati di ciascuna stazione in un unico array ed il risultato è la creazione di soli tre array: *ToConsolataArr, ToLingottoArr, ToRubinoArr*.

Anche nel caso dei file relativi a pioggia, vento e temperatura è stata eseguita una selezione dei campi. Prima di questa operazione, però, sono state filtrate le prime dieci righe di ciascun file in modo da rimuovere informazioni superflue. Per il file relativo alla pioggia sono stati selezionati:

- il campo in posizione zero, ovvero la data di rilevamento;
- il campo in posizione uno, ovvero i millimetri totali di pioggia caduti nella giornata.

Poi, per il file relativo al vento, sono stati selezionati:

- il campo in posizione zero, ovvero la data di rilevamento;
- il campo in posizione uno, ovvero la velocità media del vento durante l'arco della giornata.

Infine, per il file relativo alla temperatura, sono stati selezionati:

- il campo in posizione zero, ovvero la data di rilevamento;
- il campo in posizione uno, ovvero la temperatura media del giorno in esame.

Il risultato è la creazione di tre array, *ToPioggiaArr*, *ToVentoArr* e *ToTemperaturaArr*, in cui ciascun elemento è ancora una volta una coppia di valori.

A questo punto arriviamo ad avere tre array per il PM10, uno per stazione, e tre array per i fattori meteorologici. Si osserva che le dimensioni degli array non sono uguali; quindi è necessario prendere come riferimento *ToConsolataArr* in cui si ha la certezza che tutti i giorni siano riportati correttamente. Ciascun altro array viene percorso confrontando il campo relativo alla data di rilevamento con il corrispettivo di *ToConsolataArr* e, se avviene un match, viene salvato il valore di interesse in un altro array (*DateArrFin, ToConsolataArrFin, ToLingottoArrFin, ToRubinoArrFin, TempValorePresenteFin, PioggiaValorePresenteFin, VentoValorePresenteFin*). Un particolare accorgimento riguarda l'eliminazione, se presente, di uno spazio nella stringa relativa al valore di interesse, in modo da poter eseguire una conversione in float (questo viene fatto per tutti gli array escluso quello relativo alle date). Nel caso il match per data non avvenga con successo, in quanto in uno degli array manca la data relativa ad un particolare giorno, viene aggiunto un valore vuoto.

L'efficacia delle operazioni appena svolte è testimoniata dalla lunghezza degli array pre e post intervento. Il risultato da terminale è il seguente:

*————————-PRE-CREANING————————-*

*ToPioggiaArr: 2347*

*ToVentoArr: 2349*

*ToTemperaturaArr: 2349*

*ToConsolataArr: 2353*

*ToLingottoArr: 2353*

*ToRubinoArr: 2353*

*————————-POST-CREANING————————-*

*DateArrFin: 2353*

*ToConsolataArrFin: 2353*

*ToLingottoArrFin: 2353*

*ToRubinoArrFin: 2353*

*TempValorePresenteFin: 2353*

*PioggiaValorePresenteFin: 2353*

*VentoValorePresenteFin: 2353*

Analizzando i dati, però, si osservano valori assenti in alcuni giorni. Per questa ragione si è pensato di eseguire una interpolazione associando inizialmente ciascun array con un corrispettivo Numpy, creando poi un dataframe Pandas ed utilizzando, infine, la funzione *interpolate()*. Es: [0, , , 6] -> [0, 2, 4, 6]

Ricordando lo scopo di avere tre array per il particolato, cioè eliminare eventuali outlier causati da una misurazione errata, si è voluto creare un ulteriore array in cui a ciascun giorno venga associato il valore mediano fra i tre valori provenienti dalle stazioni fisse. Per fare ciò è stato usato il modulo Python statistics.

A questo punto tutti gli array hanno stessa lunghezza e dati completi, quindi si può procedere con la creazione del dataset finale. Questo è un file `.csv` denominato *DatasetFin* che, una volta creato e popolato, si presenta come in Tabella 2.6.

| **DataRilevamento** | **PM10TorinoConsolata** | **PM10TorinoLingotto** |
|---|---|---|
| 01/01/2017 | 89.0 | 108.0 |
| **PM10TorinoRubino** | **PM10Median** | **TMedGiardiniReali** |
| 117.0 | 108.0 | 1.9 |
| **PioggiaGiardiniReali** | **VelMedVentoViaDellaConsolata** | |
| 0.0 | 0.7 | |

**Tabella 2.5:** Esempio di dati relativi al dataset finale delle stazioni fisse

Ora che è il dataset è pronto si può procedere con l'analisi e la creazione del modello predittivo.

## 2.3 Analisi dei dati

L'analisi dei dati si suddivide in due file che permettono di distinguere lo studio relativo alla correlazione tra inquinamento e fattori meteorologici e lo studio sulle serie temporali relativo al solo PM10. Per ciascuna macroanalisi è stato creato un notebook Jupyter, il quale permette di creare e condividere documenti testuali interattivi, contenenti oggetti, quali equazioni, grafici e codice sorgente eseguibile. Quindi, è possibile eseguire e testare blocchi di codice in modo semi-indipendente gli uni dagli altri.

### 2.3.1 Correlazione inquinamento – fattori meteorologici

In questa sezione si vuole capire se e quanto pioggia, vento e temperatura influenzano i livelli di particolato nell'aria di Torino. La prima operazione che viene svolta è la lettura dal dataset creato in precedenza e la creazione di un array per ciascuna colonna del file. Viene poi creata una matrice che, di fatto, è un array *numpy* in cui ciascun elemento è uno degli array appena creati. Quindi la prima riga sarà costituita dai valori di PM10 della stazione di Torino Consolata, la seconda dai valori della stazione di Torino Lingotto, e così via. I valori della matrice vengono, poi, uniformati nella forma, cioè tutti i valori vengono ad avere al più due cifre decimali. La creazione della matrice favorisce il raggiungimento dell'obiettivo di questa analisi, cioè lo studio sulla correlazione fra i campi del dataset. Ciò che si ottiene in seguito all'esecuzione della funzione che calcola il valore di correlazione, e alla stampa del risultato, è visibile in Figura 2.2.

Per comprendere il risultato ottenuto è necessario fornire una definizione al concetto di correlazione fra due variabili. Definiamo il coefficiente di correlazione come una misura specifica usata nell'analisi della correlazione per quantificare la forza della relazione lineare tra due variabili. Nel caso di due variabili, la formula confronta la distanza di ogni punto (dato) dalla media della variabile, definendo, così, quanto la relazione tra le due variabili si posizionerebbe vicino a una linea immaginaria tracciata tra i dati. ll coefficiente di correlazione è un valore privo di unità di misura e compreso tra -1 e 1. Più ci si avvicina a zero e più la correlazione è debole, mentre un valore negativo è indice di una correlazione negativa, in cui il valore di una variabile tende ad aumentare quando l'altra diminuisce.

Fatta questa premessa, dalla Figura 2.2 si possono dedurre le seguenti conclusioni:

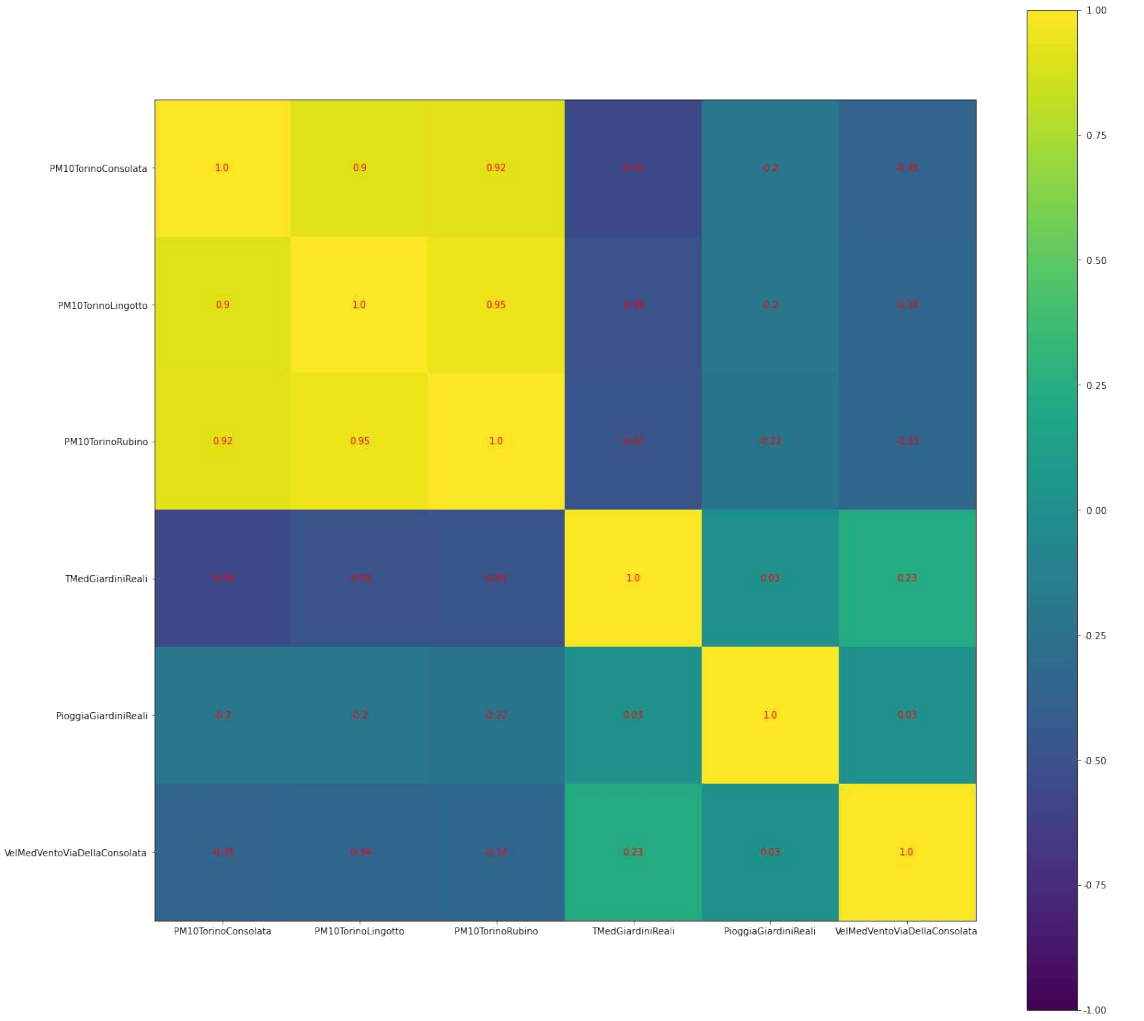


**Figura 2.2:** Grafico di correlazione inquinamento - fattori meteorologici

- I valori di PM10 delle stazioni fisse sono fortemente correlati positivamente (maggiore di 0.9 in tutti i casi); ciò rappresenta una conferma che i valori sono affidabili e coerenti tra loro.
- La temperatura media giornaliera ha una correlazione abbastanza importante, ma negativa, con il PM10 (-0.55, -0.49, -0.47); dunque in inverno ci saranno valori di particolato più alti, mentre in estate valori di particolato più bassi.
- La pioggia ha una leggera correlazione negativa con il PM10 (-0.2, -0.22, -0.22); dunque la pioggia potrebbe essere legata a valori di PM10 leggermente più bassi.
- Il vento ha una modesta correlazione negativa con il PM10 (-0.35, -0.34, -0.33); dunque il vento potrebbe essere legato a valori di PM10 leggermente più bassi. Questo punto verrà approfondito successivamente.

Ora si vuole visualizzare i punti in un grafico che abbia come ascissa il fattore meteorologico di interesse (temperatura o vento o pioggia) e come ordinata il valore di PM10. Lo scopo è vedere la distribuzione dei punti nel grafico per confermare i risultati sulla correlazione, per individuare pattern interessanti e per comprendere se la stagione dell'anno possa influire su alcuni valori.

Per perseguire quest'ultimo punto è stata costruita una funzione che riceve in input una data e restituisce in output la stagione. Grazie alla costruzione del dataset visto in precedenza è possibile leggere una riga per volta ed individuare, dal primo valore, la data. Ottenuta la data, viene eseguita la conversione in stagione e vengono salvati i valori dei successivi campi di interesse nei corrispettivi array relativi a quella specifica stagione. Gli array risultanti da tale operazione sono i seguenti:

- *PM10MedInv*: valori di PM10 (mediana tra le stazioni) registrati in inverno;
- *PM10MedPri*: valori di PM10 (mediana tra le stazioni) registrati in primavera;
- *PM10MedEst*: valori di PM10 (mediana tra le stazioni) registrati in estate;
- *PM10MedAut*: valori di PM10 (mediana tra le stazioni) registrati in autunno;
- *TMedGiardiniRealiInv*: valori di temperatura media giornaliera registrati in inverno;
- *TMedGiardiniRealiPri*: valori di temperatura media giornaliera registrati in primavera;
- *TMedGiardiniRealiEst*: valori di temperatura media giornaliera registrati in estate;
- *TMedGiardiniRealiAut*: valori di temperatura media giornaliera registrati in autunno;
- *PioggiaGiardiniRealiInv*: valori di pioggia registrati in inverno;
- *PioggiaGiardiniRealiPri*: valori di pioggia registrati in primavera;
- *PioggiaGiardiniRealiEst*: valori di pioggia registrati in estate;
- *PioggiaGiardiniRealiAut*: valori di pioggia registrati in autunno;
- *VelMedVentoViaDellaConsolataInv*: valori sulla velocità del vento registrati in inverno;
- *VelMedVentoViaDellaConsolataPri*: valori sulla velocità del vento registrati in primavera;
- *VelMedVentoViaDellaConsolataEst*: valori sulla velocità del vento registrati in estate;
- *VelMedVentoViaDellaConsolataAut*: valori sulla velocità del vento registrati in autunno.

Il primo grafico che si ottiene, Figura 2.3, riguarda gli array sul PM10 ed i corrispettivi sulla temperatura media giornaliera.

La prima cosa che salta all'occhio, conferma dell'analisi precedente, è che a temperature basse corrispondono valori di particolato alti, o comunque picchi di inquinamento più alti, mentre a temperature alte corrispondono livelli di inquinamento più bassi. Nella stagione estiva, infatti, non ci sono mai picchi di inquinamento alti, risultando, così, la stagione migliore dal punto di vista salutare. L'inverno, invece, risulta la stagione peggiore. Questo fenomeno prende il nome di inversione termica invernale. In presenza di giornate più corte, irraggiamento solare più debole e temperature più basse il terreno non riesce a riscaldarsi a sufficienza per riscaldare, a sua volta, l'aria. In questo modo l'aria a contatto con il terreno si raffredda rapidamente e raggiunge temperature inferiori rispetto a quella sovrastante, generando una sorta di cappa che impedisce alle sostanze inquinanti di disperdersi negli strati superiori dell'atmosfera.

Il secondo grafico che analizziamo, Figura 2.4, riguarda gli array sul PM10 ed i corrispettivi sui millimetri di pioggia giornalieri.

Dal grafico si può notare una tendenza ad avere relativamente poca pioggia nella stagione invernale e la presenza di giornate con tantissima pioggia in estate ed autunno. La prima è una

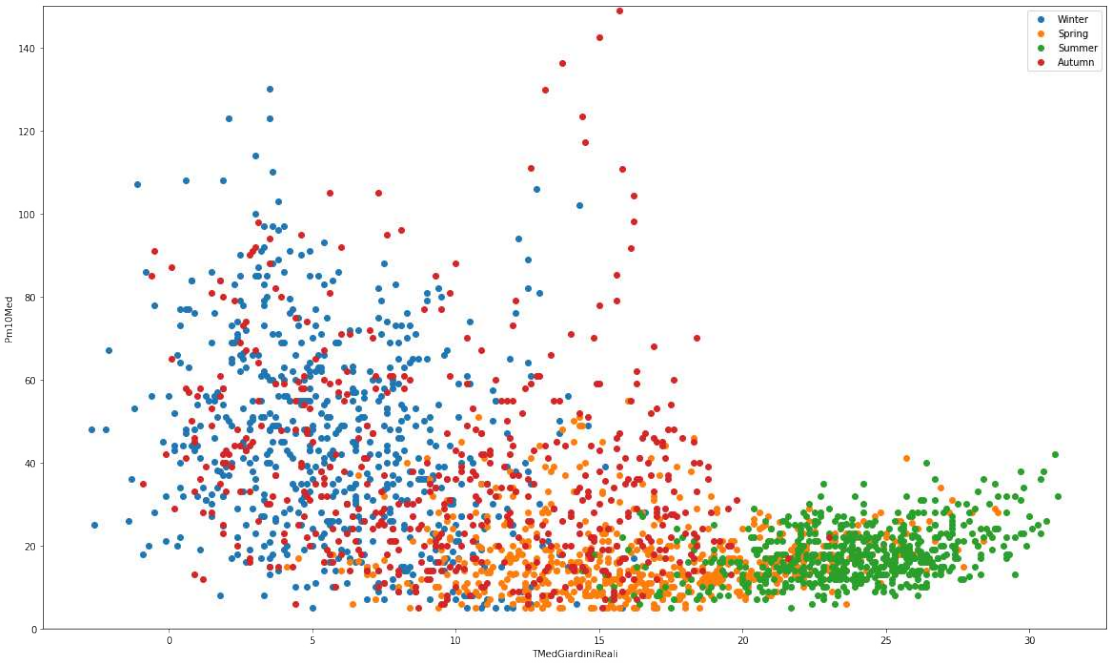


**Figura 2.3:** Grafico tra il PM10 e la temperatura media giornaliera

caratteristica del clima di Torino, con inverni freddi ma con scarse precipitazioni, che, invece, si concentrano nelle altre stagioni. La seconda considerazione può essere, semplicemente, spiegata con la presenza di temporali nelle stagioni più calde. Come avevamo visto anche nel grafico della correlazione, non c'è una dipendenza così stretta tra pioggia e PM10. Si può, però, notare come i livelli di PM10 alti siano concentrati nei giorni con assenza di pioggia; quindi, un leggero effetto positivo da parte dei rovesci sull'inquinamento c'è.

Il terzo ed ultimo grafico che analizziamo, Figura 2.5, riguarda gli array sul PM10 ed i corrispettivi sulla velocità media del vento di un giorno.

I valori sono discreti in quanto la sensibilità dei sensori della stazione fissa è di 0.1m/s. Durante l'inverno c'è più variabilità sulla velocità del vento, mentre nelle altre stagioni difficilmente si superano di molto i 2 m/s di media. Si può notare un leggero andamento decrescente del grafico, ma questo non è particolarmente significativo. È interessante notare come nell'intorno di 2 sia presente un calo significativo dei livelli di inquinamento. Questo potrebbe essere spiegato dalla difficoltà di "pulire l'aria" da parte del vento sotto certe velocità. Quando, invece, si raggiungono valori superiori ai 2 m/s sembra che si creino le condizioni per spazzare via il particolato.

Da queste analisi si può giungere alla conclusione che tra le stazioni fisse, che misurano il PM10, non vi sono particolari differenze. Da un certo punto di vista ciò ci garantisce sull'affidabilità dei valori estratti, dall'altra non ci permette di capire se alcune zone sono particolarmente più inquinate di altre, motivo per cui l'estrazione di dati puntuali con Arduino gioca un ruolo chiave. L'altra considerazione è che le temperature, e di conseguenza le stagioni, giocano un ruolo importante sui livelli di inquinamento. La pioggia ed il vento, in misura minore, possono contribuire, soprattutto superate delle soglie, all'abbassamento dei livelli di particolato.

### 2.3.2 Serie temporali

Grazie alle analisi precedenti si è compreso come il PM10 sia influenzato da fattori meteorologici, ma ciò non ci permette di comprendere l'andamento e la stagionalità di tali

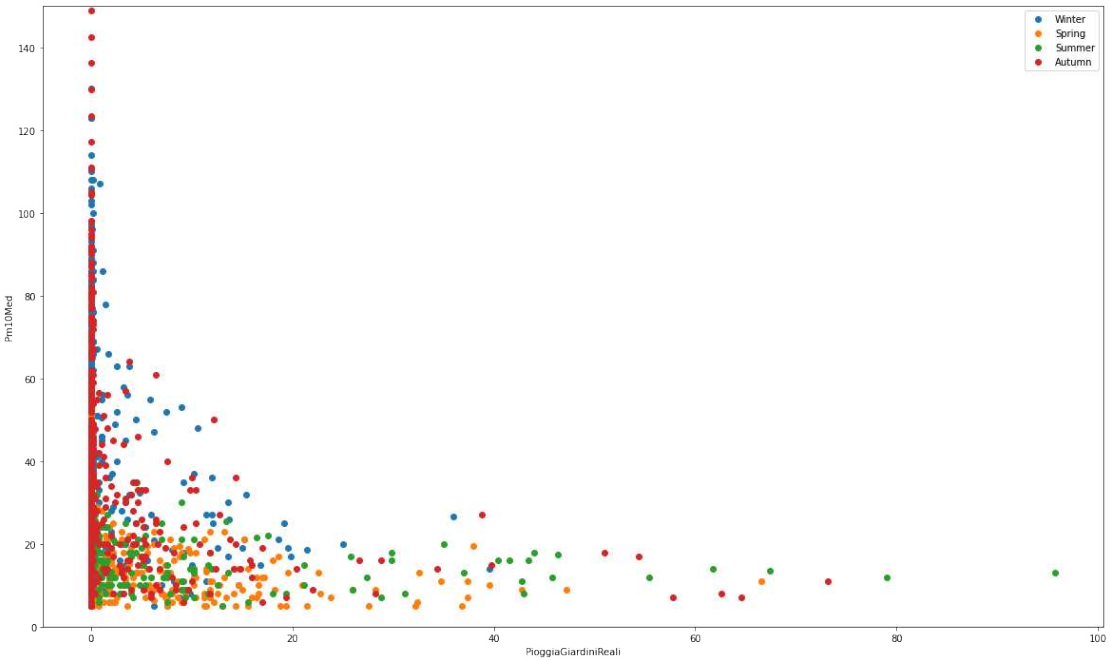


**Figura 2.4:** Grafico tra il PM10 ed i millimetri di pioggia giornalieri

valori. Inoltre, si vuole creare un modello predittivo per i valori di particolato.

Il primo obiettivo di questa sezione è quello di approfondire la questione tramite un'analisi basata su serie temporali.

Per efficientare le operazioni di analisi e di creazione del modello si è deciso di leggere il dataset creato precedentemente e, sulla base di esso, costruire un dataset con valori settimanali, ricavati dalla media dei valori di ciascuna settimana (ad eccezione della data).

Una volta creato tale dataset viene letto e creato un dataframe Pandas, cioè una struttura dati tabulare bidimensionale con dimensione variabile, potenzialmente eterogenea e con assi etichettati. Vediamo il risultato della funzione *df.head()*, che mostra le prime righe del dataframe appena creato, nella Figura 2.6.

Come possiamo osservare, gli indici del dataframe sono numeri incrementali; però vorremmo che gli indici fossero le date di inizio settimana. Per un corretto riconoscimento del campo DataRilevamento lo convertiamo in tipo datetime64, specificando il formato %d/%m/%Y, e lo settiamo come indice. Tale operazione produce il risultato osservabile in Figura 2.7.

Come si può notare dalla figura precedentemente citata, il campo settato come indice viene visualizzato in una riga differente dagli altri campi. La funzione *df.index*, ancora una volta, ci permette di avere una panoramica dei valori, del tipo e del numero di elementi. Il risultato prodotto è riportato in Figura 2.8.

Per verificare i valori del primo ed ultimo giorno del dataframe vengono usate, rispettivamente, le funzioni *df.iloc[0]* e *df.iloc[-1]* ottenendo quanto presente in Figura 2.9 e 2.10.

Può rivelarsi molto utile esplorare ulteriormente i dati a disposizione con alcune metriche, come il numero di elementi, la media, la deviazione standard, i valori minimo e massimo, ed i vari percentili che ci permettono di capire la distribuzione dei valori. Utilizzando la funzione pandas *statistics* si ottiene il risultato di Figura 2.11.

Si può notare come i valori di PM10 oscillino tra 151 µg/m³, valore altissimo e pericoloso per la salute, ed 8 µg/m³, con una media di quasi 30 µg/m³. Confrontando quest'ultimo valore con la media italiana di 18.9 µg/m³ viene confermato ancora una volta il problema

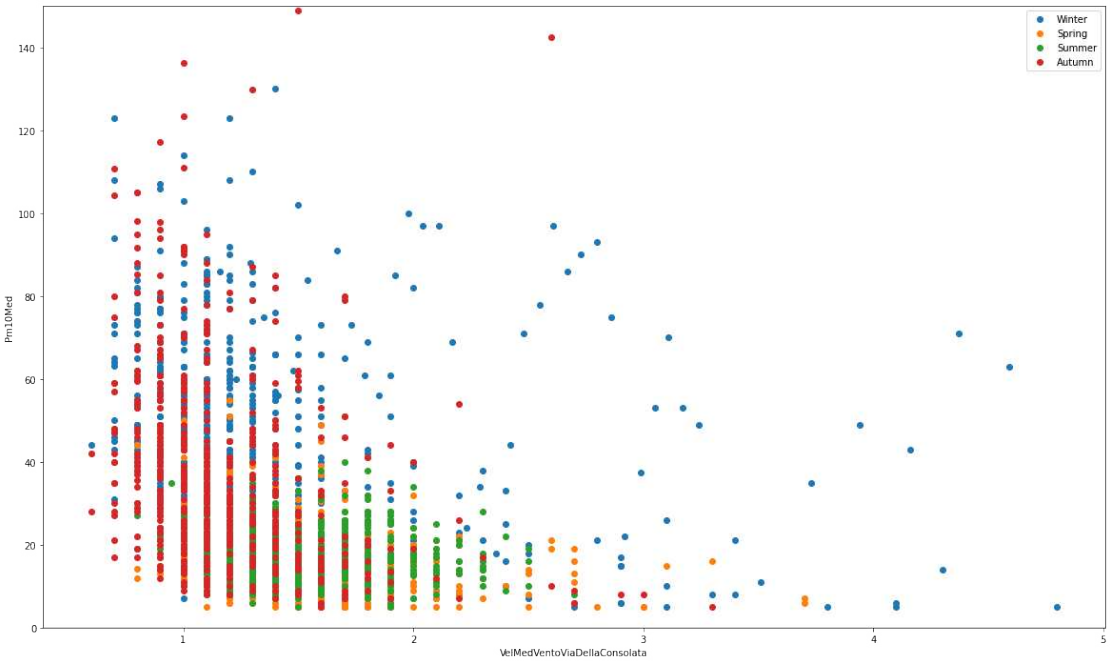


**Figura 2.5:** Grafico tra il PM10 e la velocità media del vento giornaliera

| | DataRilevamento | PM10Median | TMedGiardiniReali | PioggiaGiardiniReali | VelMedVentoViaDellaConsolata |
|---|---|---|---|---|---|
| 0 | 01/01/2017 | 60.2857 | 1.6286 | 0.1000 | 1.0571 |
| 1 | 08/01/2017 | 60.8571 | 0.6143 | 1.1000 | 1.4714 |
| 2 | 15/01/2017 | 48.7143 | 0.7857 | 0.0429 | 0.9571 |
| 3 | 22/01/2017 | 72.0000 | 3.0143 | 0.0429 | 1.0714 |
| 4 | 29/01/2017 | 71.7143 | 4.3286 | 1.1143 | 1.0571 |

**Figura 2.6:** Dataframe iniziale rappresentativo dei dati settimanali di PM10 e fattori meteorologici

dell'inquinamento nelle grandi città del Nord Italia. Il valore 50% ci mostra come, tendenzialmente, i valori di particolato siano più bassi della media, la quale è infuenzata da picchi. Le temperature medie giornaliere, invece, variano tra i -0.89°C ed i 29.79°C, in linea con quanto ci si aspetta. Il valore medio è 14.09°C, in linea con il valore del 50° percentile; dunque, si ha una distribuzione centrata proprio sul valore medio. Un discorso diverso si ha per la pioggia in quanto sono presenti molti valori pari a 0, dimostrato dallo 0.03 per il 25° percentile. Il valore massimo è di 21.94mm di pioggia. Il vento ha una distribuzione con un valore mediano in linea con la media di 1.40 m/s; tale valore risulta essere piuttosto basso.

Dopo aver analizzato i dati secondo indici piuttosto semplici si vuole comprendere dal punto di vista statistico se sono presenti trend, autocorrelazione o stagionalità nei dati di PM10.

Il primo indice che andiamo a valutare è l'ADF. Il test Augmented Dickey Fuller (ADF Test) è un test statistico utilizzato per verificare se una determinata serie temporale è stazionaria o meno. È uno dei test statistici più comunemente usati per questo scopo; per comprenderne il significato facciamo delle premesse. Nella previsione delle serie temporali con ARIMA, il primo passaggio consiste nel determinare il numero di differenziazioni necessarie per rendere stazionaria la serie. Questo test appartiene alla categoria di test per la stazionarietà denominata 'Unit Root Test', in cui si dice che la 'Unit Root' esiste in una serie temporale dal valore di alfa = 1 prendendo in esame l'equazione:

| DataRilevamento | PM10Median | TMedGiardiniReali | PioggiaGiardiniReali | VelMedVentoViaDellaConsolata |
|---|---|---|---|---|
| 2017-01-01 | 60.2857 | 1.6286 | 0.1000 | 1.0571 |
| 2017-01-08 | 60.8571 | 0.6143 | 1.1000 | 1.4714 |
| 2017-01-15 | 48.7143 | 0.7857 | 0.0429 | 0.9571 |
| 2017-01-22 | 72.0000 | 3.0143 | 0.0429 | 1.0714 |
| 2017-01-29 | 71.7143 | 4.3286 | 1.1143 | 1.0571 |

**Figura 2.7:** Dataframe con la data come indice

```
DatetimeIndex(['2017-01-01', '2017-01-08', '2017-01-15', '2017-01-22',
               '2017-01-29', '2017-02-05', '2017-02-12', '2017-02-19',
               '2017-02-26', '2017-03-05',
               ...
               '2023-04-02', '2023-04-09', '2023-04-16', '2023-04-23',
               '2023-04-30', '2023-05-07', '2023-05-14', '2023-05-21',
               '2023-05-28', '2023-06-04'],
              dtype='datetime64[ns]', name='DataRilevamento', length=336, freq=None)
```

**Figura 2.8:** Riassunto dei valori degli indici del dataframe

$$Y_t = \alpha Y_{t-1} + \beta X_e + \epsilon \tag{2.3.1}$$

$Yt$ è il valore della serie temporale al tempo *'t'* e $Xe$ è una variabile esogena. Il numero di 'Unit Root' contenute nella serie corrisponde al numero di operazioni di derivazione necessarie per rendere stazionaria la serie. Il test ADF è fondamentalmente un test di significatività statistica e ciò significa che esiste un test di ipotesi coinvolto, con un'ipotesi nulla ed una alternativa. Di conseguenza, viene calcolata una statistica di test e vengono riportati i p-value. Scendendo nel dettaglio, nel test Dickey-Fuller abbiamo che l'ipotesi nulla è $\alpha$=1 nella seguente equazione ($\alpha$ è il coefficiente del primo lag della Y).

$$y_t = c + \beta t + \alpha Y_{t-1} + \Phi \Delta Y_{t-1} + e_t \tag{2.3.2}$$

Il test ADF espande l'equazione del test Dickey-Fuller per includere il processo regressivo di ordine elevato nel modello. Si arriva, così, alla seguente formula:

$$y_t = c + \beta t + \alpha Y_{t-1} + \Phi_1 \Delta Y_{t-1} + \Phi_2 \Delta Y_{t-2} .. + \Phi_p \Delta Y_{t-p} + e_t \tag{2.3.3}$$

Più è negativo il valore di ADT e più forte è il rifiuto dell'ipotesi che esista una radice unitaria ad un certo livello di confidenza (Tabella 2.6).

| Sample size | 1% no trend | 5% no trend | 1% trend | 5% trend |
|---|---|---|---|---|
| T=25 | -3.75 | -3.00 | -4.38 | -3.60 |
| T=50 | -3.58 | -2.93 | -4.15 | -3.50 |
| T=100 | -3.51 | -2.89 | -4.04 | -3.45 |
| T=250 | -3.46 | -2.88 | -3.99 | -3.43 |
| T=500 | -3.44 | -2.87 | -3.98 | -3.42 |
| T=$\infty$ | -3.43 | -2.86 | -3.96 | -3.41 |

**Tabella 2.6:** Esempio di dati relativi al dataset finale delle stazioni fisse

```
PM10Median                      60.2857
TMedGiardiniReali                1.6286
PioggiaGiardiniReali             0.1000
VelMedVentoViaDellaConsolata     1.0571
Name: 2017-01-01 00:00:00, dtype: float64
```

**Figura 2.9:** Dati della prima riga del dataframe

```
PM10Median                      13.0000
TMedGiardiniReali               20.3286
PioggiaGiardiniReali             6.4571
VelMedVentoViaDellaConsolata     1.3000
Name: 2023-06-04 00:00:00, dtype: float64
```

**Figura 2.10:** Dati dell'ultima riga del dataframe

Una volta che si hanno i valori di riferimento, si possono confrontare questi ultimi con i risultati prodotti, Figura 2.12.

Il valore $p$ è molto inferiore al livello di significatività di 0,05; quindi possiamo rifiutare l'ipotesi nulla e ritenere che la serie sia stazionaria.

Prima di continuare con lo studio dei dati a disposizione è utile visualizzarli per analizzarne l'andamento (Figura 2.13).

Dalla figura vengono confermate molte delle considerazioni che erano state esposte precedentemente. In particolare, si può notare l'andamento periodico del PM10 con valori bassi durante la stagione calda e valori alti in quella fredda. Un andamento quasi opposto riguarda le temperature, ed in misura minore le piogge. Ovviamente queste ultime non hanno un andamento regolare ma si osservano meno precipitazioni nel periodo invernale. La velocità del vento ha un andamento simile a quello delle temperature, ma con più irregolarità e differenze meno marcate tra i valori minimi e quelli massimi. Inoltre, sembra anticipato il periodo di minimo di un paio di mesi.

Per verificare se ci sono variazioni improvvise tra i valori giornalieri di PM10 viene sfruttata la funzione *diff* di *pandas*, la quale restituisce un dataframe contenente la differenza tra i valori di ogni riga e, per impostazione predefinita, la riga precedente. La riga con cui eseguire il confronto può essere specificata con il parametro *Periods*, ma useremo l'impostazone di default. Una cosa analoga verrà fatta per il risultato dell'operazione precedente, in modo da verificare se tra le variazioni vi siano, a loro volta, differenze importanti. Il risultato è osservabile in Figura 2.14.

Dal grafico precedente si può dedurre che le variazioni più importanti siano presenti quando i valori di PM10 sono più alti. Questo periodo corrisponde ai mesi in cui le temperature sono mediamente più basse, l'attività industriale è a pieno regime e lo spostamento con auto, e simili, viene preferito. Questo grafico ci permette di verificare, anche, per quale "ordine delle differenze" il grafico diventa stazionario.

Dopo aver compreso meglio l'andamento temporale dei valori di particolato, e la relazione con i fattori meteorologici, ci si concentra sulla costruzione di un modello predittivo che prenda in considerazione il trend e la stagionalità.

Per costruire un modello predittivo SARIMAX è necessario studiare e determinare i valori che compongono un modello ARIMA, in quanto la S indica che viene presa in esame anche la stagionalità mentre la X, non sfruttata nel modello finale, indica che vengono valutate altre variabili (variabili esogene, come la temperatura) che possono influenzare il valore della variabile di interesse.

| | PM10Median | TMedGiardiniReali | PioggiaGiardiniReali | VelMedVentoViaDellaConsolata |
|---|---|---|---|---|
| count | 336.000000 | 336.000000 | 336.000000 | 336.000000 |
| mean | 29.600768 | 14.086821 | 2.286906 | 1.404146 |
| std | 19.152947 | 7.814449 | 3.455523 | 0.310031 |
| min | 8.000000 | -0.885700 | 0.000000 | 0.828600 |
| 25% | 15.767850 | 6.982125 | 0.028600 | 1.185700 |
| 50% | 23.035750 | 13.778550 | 0.671400 | 1.400000 |
| 75% | 39.571400 | 20.839275 | 3.164300 | 1.571400 |
| max | 151.632900 | 29.785700 | 22.628600 | 3.578600 |

**Figura 2.11:** Informazioni statistiche sui valori dei campi del dataframe

```
ADF Statistics: -5.074447
p-value: 0.000016
Critial Values:
   1%, -3.450632157720528
Critial Values:
   5%, -2.870474482366864
Critial Values:
   10%, -2.5715301325443787
```

**Figura 2.12:** Valori del test ADF e p-value relativi ai dati di PM10

ARIMA è l'acronimo di AutoRegressive Integrated Moving Average ed è una tipologia di modelli appartenenti alla famiglia dei processi lineari non stazionari.

Un modello ARIMA presenta tre parametri di interesse ($p$, $d$ e $q$) che sono numeri interi non negativi.

- $p$ rappresenta il numero di parametri autoregressivi;
- $d$ rappresenta l'ordine di differenziazione;
- $q$ rappresenta il numero di parametri a media mobile.

La componente autoregressiva del modello ARIMA è rappresentata da $AR(p)$, con il parametro $p$ che determina il numero di lag che vengono usati. Se impostiamo il parametro $p$ a 0 otteniamo il rumore bianco. Ogni dato viene campionato da una distribuzione con media 0 e varianza $\sigma^2$. Ciò si traduce in una sequenza di numeri casuali che non possono essere previsti. Se, invece, impostiamo il parametro $p$ ad 1, prendiamo in considerazione il timestamp precedente, ma regolato da un moltiplicatore, e poi aggiungiamo il rumore bianco. Aumentare ulteriormente il parametro $p$ significa solo andare più indietro e aggiungere più timestamp regolati dai propri moltiplicatori. Tornando troppo indietro è più probabile che dovremmo utilizzare parametri aggiuntivi come la media mobile (MA), per compensare.

$MA(q)$ è il modello a media mobile e $q$ è il numero di termini, di errore, di previsione ritardata. Il modello esprime il valore attuale come combinazione lineare della media della serie, del termine di errore presente e dei termini di errore passati. Quindi, in un modello $MA(1)$, la nostra previsione è un termine costante più il termine del rumore bianco precedente moltiplicato per un moltiplicatore, sommato poi al rumore bianco corrente.

$I(d)$ rappresenta la differenziazione delle osservazioni grezze per consentire alle serie temporali di diventare stazionarie. In parole più semplici, i valori dei dati vengono sostituiti dalla differenza tra i valori dei dati e i valori precedenti.

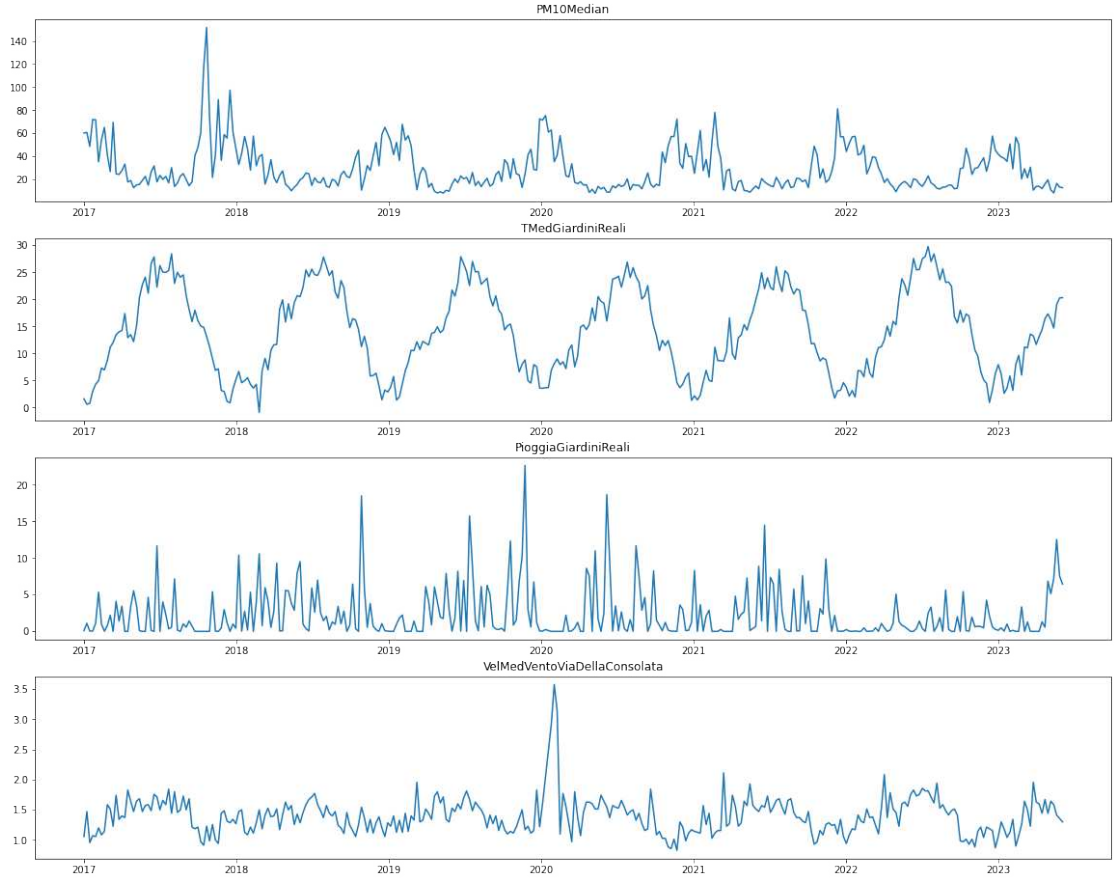


**Figura 2.13:** Visualizzazione andamento temporale PM10, vento, pioggia e temperatura

Dopo questa doverosa premessa si determinano i valori che costituiscono il modello ARIMA.

Il primo grafico che viene analizzato è riportato in Figura 2.15 e vengono graficati i valori delle funzioni di autocorrelazione. In particolare, viene riportata la serie originale, il primo ed il secondo ordine di differenziazione.

Viene scelto $d = 0$ in quanto il grafico di autocorrelazione per il primo ordine di differenza va nella zona negativa troppo velocemente. Questo vale a maggior ragione per il secondo ordine di differenza.

Per calcolare il valore di $p$ per ARIMA viene graficata, in Figura 2.16, l'autocorrelazione parziale.

Escluso il primo punto, ce ne sono altri che superano il livello di significatività, seppur con valori negativi. Nell'ottica di scegliere un modello semplice viene selezionato $p = 1$. Stesso discorso vale per il grafico sull'autocorrelazione in cui, escluso il primo punto, sono presenti altri due punti che superano il livello di significatività e conseguentemente settiamo $q = 2$.

Lanciando l'algoritmo "auto_arima" vengono testati diversi valori di $p, d, q$ e vengono selezionati coloro che portano ad un AIC più piccolo (da non considerare i valori in cui compare la scritta Intercept). L'AIC fornisce una misura della qualità della stima di un modello statistico tenendo conto sia della bontà di adattamento che della complessità del modello. Vediamo il risultato ottenuto nelle Figure 2.17 e 2.18.

L'algoritmo ci conferma quanto calcolato precedentemente ed otteniamo ARIMA(1,0,2).

A questo punto viene creato il modello ARIMA sulla base del dataframe a nostra disposizione e dei valori $p, d, q$ precedentemente calcolati. In Figura 2.19 si osservano i risultati

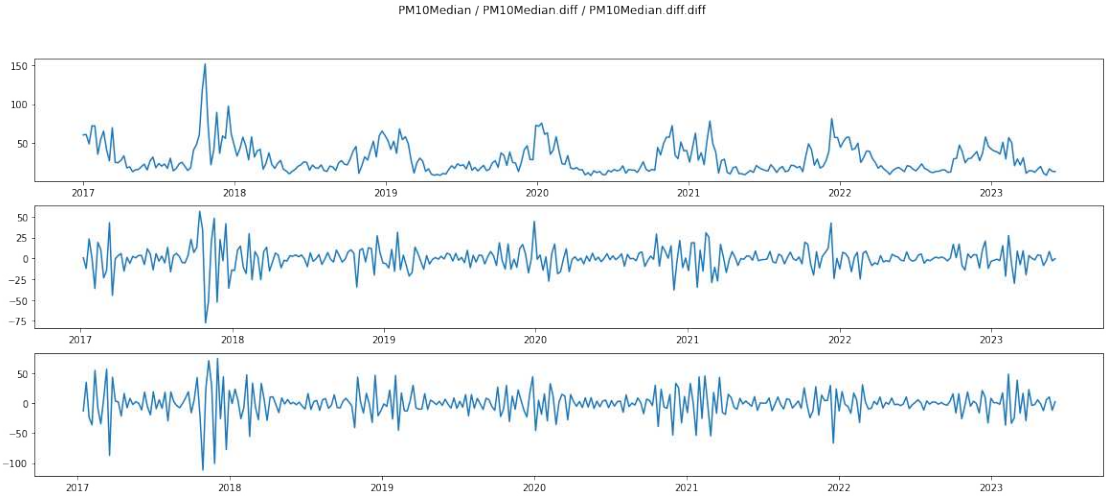


**Figura 2.14:** Variazione dei valori di PM10

ottenuti.

Il package *statsmodels* permette di ottenere un riassunto del modello creato; dalla figura appena citata si può notare come siano presenti gli elementi caratteristici di un ARIMA (1,0,2) ovvero una costante, un livello $L_1$, legato ad "AR", due livelli, $L_1$ ed $L_2$, legati alla componente "MA". Inoltre, i valori associati a $p > |z|$, quindi i p-value di ciascun sotto modello, sono pari a 0. Questo elemento è decisamente positivo in quanto, solitamente, si desiderano valori inferiori a 0.05.

Ora si vuole calcolare e visualizzare i residui, Figura 2.20, in modo da essere sicuri che il modello non abbia trascurato informazioni utili. Definiamo i residui come stime dell'errore sperimentale, ottenute sottraendo le risposte osservate dalle risposte previste. La risposta prevista viene calcolata dal modello scelto, dopo che tutti i parametri del modello sono stati stimati a partire dai dati sperimentali.

Non si notano pattern particolari tranne che una leggera stagionalità nei valori dei residui. Il tutto viene giustificato dalla struttura della serie di partenza caratterizzata da una forte stagionalità. Dunque, non siamo all'ottimo ma abbiamo comunque una distribuzione di densità a campana e centrata nello zero. Quest'ultimo elemento è importante in quanto, generalmente, per poter fare deduzioni valide su una regressione, bisognerebbe avere residui che seguono una distribuzione normale. Quanto svolto fino ad ora serve principalmente per individuare i valori di $p$, $d$, $q$ e per eseguire analisi di vario tipo. Sappiamo, però, che ARIMA è un modello che non tiene conto né della stagionalità né di variabili esogene; per questo sarà necessario creare un modello SARIMAX (non verranno sfruttate variabili esogene). Prima di creare il modello appena citato si eseguono analisi di vario tipo e si separano trend, stagionalità, livello di base e residui.

Troviamo una conferma di quanto appena detto dalla Figura 2.21 che confronta la predizione di ARIMA con la serie temporale originale. ARIMA riesce a predire tutto nel migliore dei modi ma, ovviamente, non tiene conto della stagionalità e per predizioni di medio-lungo periodo non riuscirebbe a replicare risultati così buoni.

Nella Figura 2.22 oltre alla serie originale, vengono visualizzati media e deviazione standard calcolati su una finestra temporale di 52 settimane.

Ora si vuole visualizzare anche la stagionalità e separare le componenti principali della serie. Il risultato è mostrato in Figura 2.23.

Le considerazioni che ne scaturiscono sono le seguenti:

- il trend è quasi assente; si osserva solo una leggera flessione tra il 2017 ed il 2019;

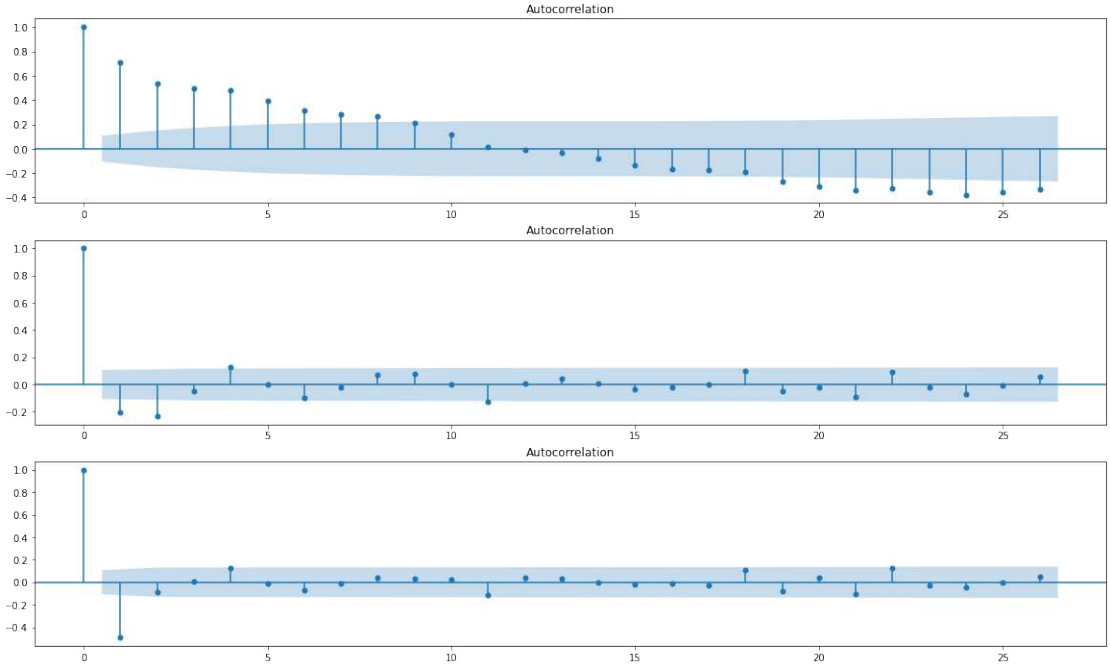


**Figura 2.15:** Grafico di autocorrelazione per il PM10

- la stagionalità forma un grafico molto pulito e ben riconoscibile;
- i residui sono piuttosto randomici (elemento decisamente positivo).

Si vuole poi dividere il dataset in due parti; una prima parte costituisce il dataset di training ed è composto da 261 elementi, circa il 78% dei dati, mentre una seconda parte costituisce il dataset con il quale andremo a testare il modello SARIMAX creato. Il dataframe di test si esaurisce il 26 dicembre 2021 come è possibile osservare nella Figura 2.24.

Infine, viene addestrato un modello SARIMAX con $p = 1$, $d = 0$, $q = 2$ (gli stessi precedentemente calcolati in ARIMA) e stagionalità di 52.179 (settimane). In Figura 2.25 viene visualizzata la serie originale e la serie predetta a partire da 261 valori su 336 totali.

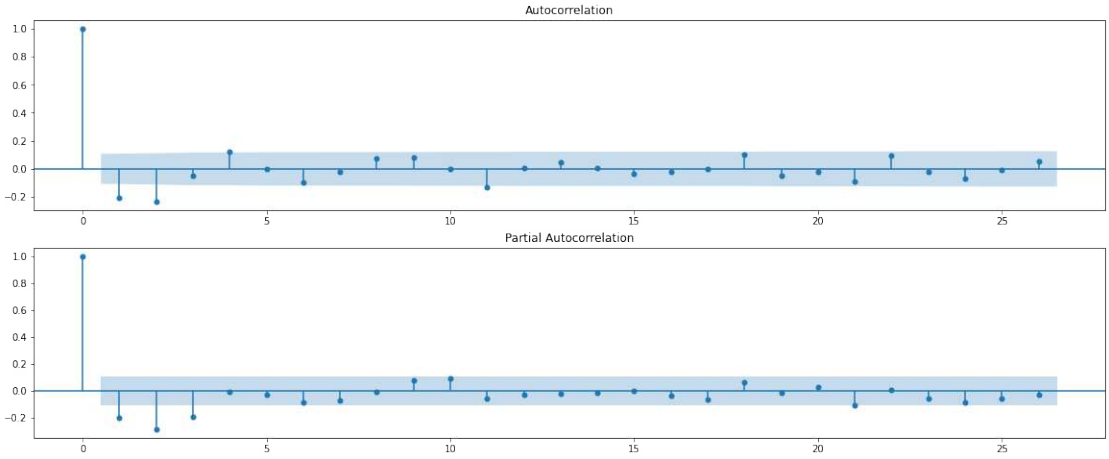


**Figura 2.16:** Grafico di autocorrelazione parziale per il PM10

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=2696.119, Time=0.48 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=2940.576, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=2705.183, Time=0.11 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=2785.993, Time=0.11 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=3349.338, Time=0.01 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=2694.204, Time=0.30 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=2744.458, Time=0.22 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=2703.203, Time=0.18 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=2695.985, Time=0.49 sec
 ARIMA(0,0,3)(0,0,0)[0] intercept   : AIC=2738.736, Time=0.32 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=2698.848, Time=0.31 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=2695.632, Time=0.91 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=2705.179, Time=0.14 sec

Best model:  ARIMA(1,0,2)(0,0,0)[0] intercept
Total fit time: 3.602 seconds
```

**Figura 2.17:** Parametri calcolati da auto arima per creazione del modello ARIMA

SARIMAX Results

| | | | |
|---|---|---|---|
| Dep. Variable: | y | No. Observations: | 336 |
| Model: | SARIMAX(1, 0, 2) | Log Likelihood | -1342.102 |
| Date: | Mon, 12 Jun 2023 | AIC | 2694.204 |
| Time: | 14:50:25 | BIC | 2713.290 |
| Sample: | 0 | HQIC | 2701.812 |
| | - 336 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| intercept | 3.0652 | 1.465 | 2.093 | 0.036 | 0.195 | 5.936 |
| ar.L1 | 0.8974 | 0.037 | 24.542 | 0.000 | 0.826 | 0.969 |
| ma.L1 | -0.2615 | 0.045 | -5.814 | 0.000 | -0.350 | -0.173 |
| ma.L2 | -0.1878 | 0.042 | -4.449 | 0.000 | -0.271 | -0.105 |
| sigma2 | 172.1433 | 8.563 | 20.103 | 0.000 | 155.360 | 188.926 |

| | | | |
|---|---|---|---|
| Ljung-Box (L1) (Q): | 0.02 | Jarque-Bera (JB): | 423.15 |
| Prob(Q): | 0.89 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 0.29 | Skew: | 1.15 |
| Prob(H) (two-sided): | 0.00 | Kurtosis: | 7.99 |

**Figura 2.18:** Analisi auto arima per la creazione del modello ARIMA

ARMA Model Results

| | | | |
|---|---|---|---|
| Dep. Variable: | PM10Median | No. Observations: | 336 |
| Model: | ARMA(1, 2) | Log Likelihood | -1342.102 |
| Method: | css-mle | S.D. of innovations | 13.120 |
| Date: | Mon, 12 Jun 2023 | AIC | 2694.204 |
| Time: | 14:50:26 | BIC | 2713.290 |
| Sample: | 01-01-2017 | HQIC | 2701.812 |
| | - 06-04-2023 | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 29.8886 | 3.761 | 7.946 | 0.000 | 22.517 | 37.261 |
| ar.L1.PM10Median | 0.8974 | 0.035 | 25.326 | 0.000 | 0.828 | 0.967 |
| ma.L1.PM10Median | -0.2615 | 0.065 | -4.049 | 0.000 | -0.388 | -0.135 |
| ma.L2.PM10Median | -0.1878 | 0.054 | -3.470 | 0.001 | -0.294 | -0.082 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| AR.1 | 1.1143 | +0.0000j | 1.1143 | 0.0000 |
| MA.1 | 1.7141 | +0.0000j | 1.7141 | 0.0000 |
| MA.2 | -3.1065 | +0.0000j | 3.1065 | 0.5000 |

**Figura 2.19:** P-value e sottocomponenti del modello ARIMA

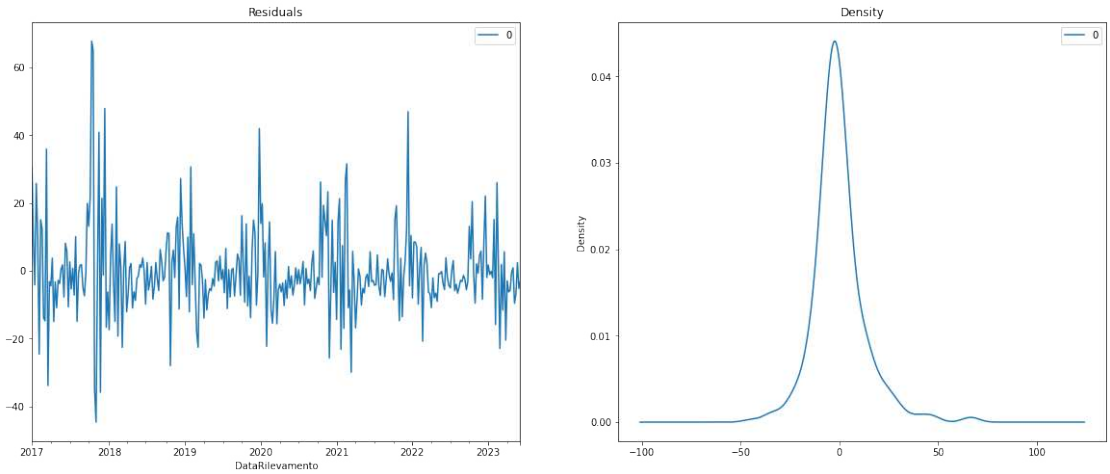


**Figura 2.20:** Visualizzazione e distribuzione dei residui

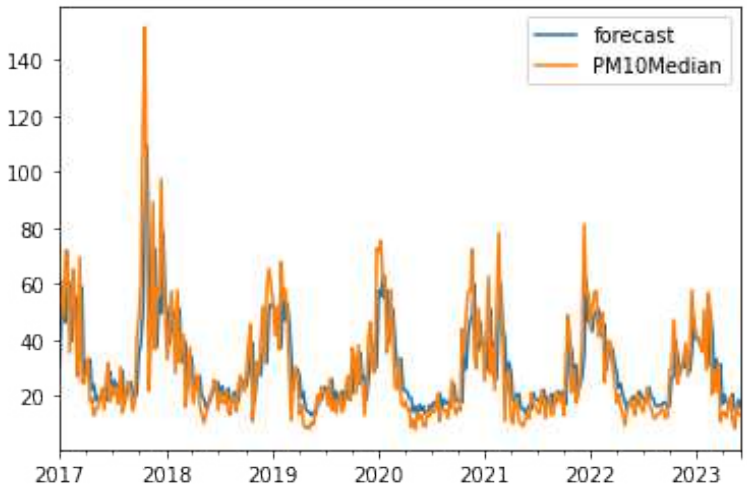


**Figura 2.21:** Predizione del modello ARIMA confrontata con i valori reali di PM10

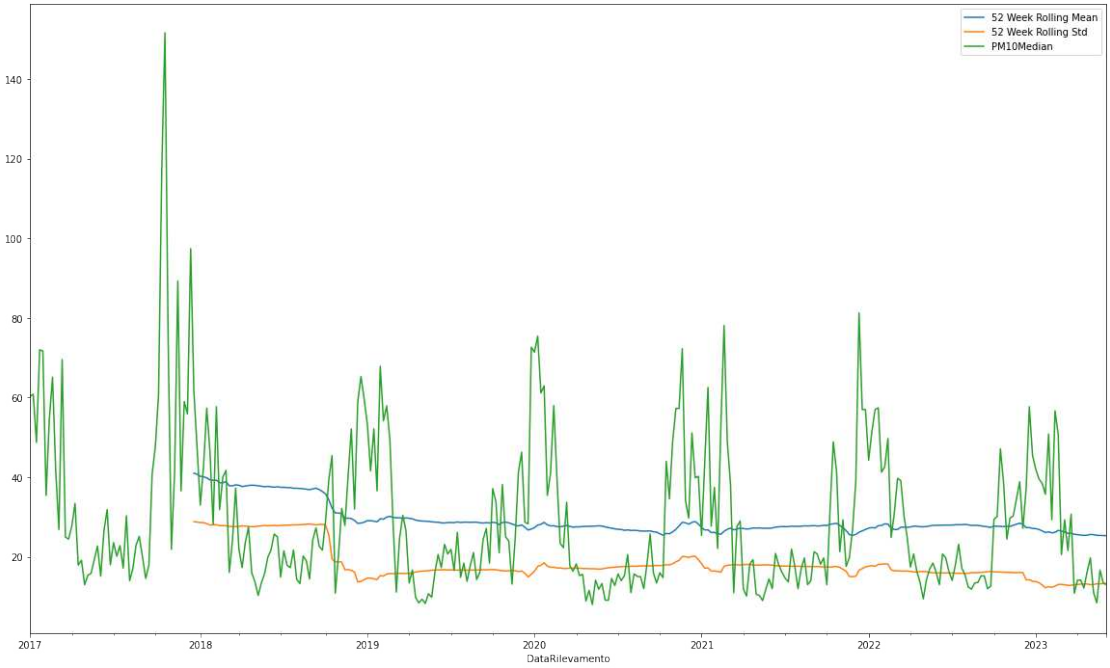


**Figura 2.22:** Visualizzazione media e deviazione standard

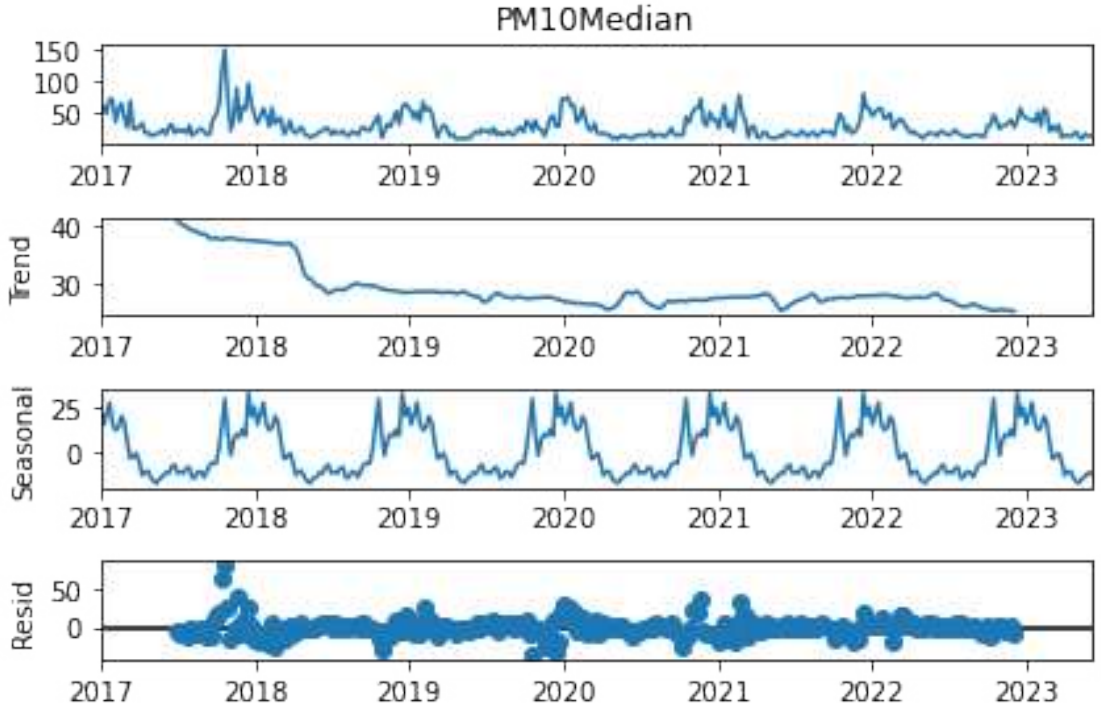


**Figura 2.23:** Grafico di separazione fra trend, stagionalità e residui

```
DatetimeIndex(['2017-01-01', '2017-01-08', '2017-01-15', '2017-01-22',
               '2017-01-29', '2017-02-05', '2017-02-12', '2017-02-19',
               '2017-02-26', '2017-03-05',
               ...
               '2021-10-24', '2021-10-31', '2021-11-07', '2021-11-14',
               '2021-11-21', '2021-11-28', '2021-12-05', '2021-12-12',
               '2021-12-19', '2021-12-26'],
              dtype='datetime64[ns]', name='DataRilevamento', length=261, freq=None)
```

**Figura 2.24:** Visualizzazione degli indici del training set

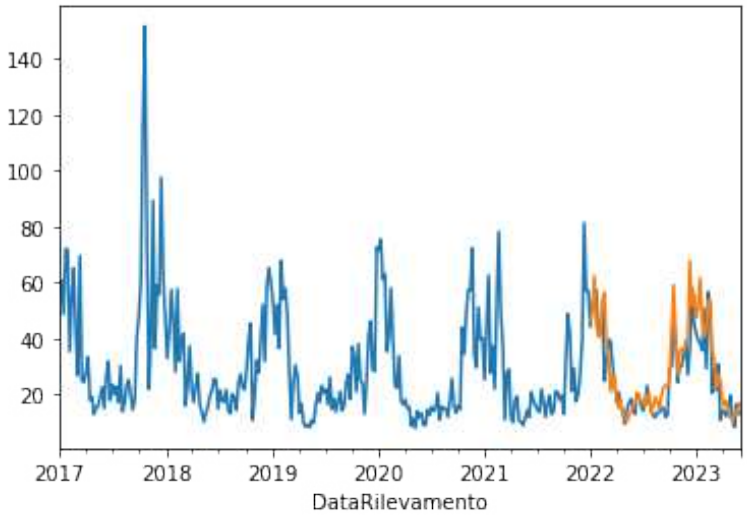


**Figura 2.25:** Grafico di predizione del modello SARIMA sul PM10

# CAPITOLO 3

## Dati puntuali acquisiti da Arduino

*Nelle città, molto spesso, il traffico si concentra solo in determinati orari e strade. Pensiamo, ad esempio, a ciò che accade nelle strade che collegano le abitazioni ai luoghi di lavoro nell'orario che va dalle 7 alle 9 circa del mattino o dalle 17 alle 19 circa di pomeriggio. Moltissime persone decidono di spostarsi in auto e si vengono a creare lunghe code, che causano consumi sia legati allo stare fermi nel traffico che alle continue accelerazioni e frenate, che sono proprio i momenti di minor efficienza dei motori termici. Di conseguenza, si creeranno aree in cui i livelli di smog sono molto alti ed aree poco trafficate e con una qualità dell'aria migliore. Per tutti coloro che scelgono di muoversi a piedi o con la bici è bene evitare le strade con livelli di particolato, ed agenti inquinanti in generale, molto alti.*

*L'obiettivo che si vuole perseguire con la creazione di un dispositivo basato su Arduino è di sfruttare un sensore di PM10 ed un GPS per raccogliere dati puntuali su tutte le strade di interesse. In questa tesi, per questione di costi e di tempi, saranno presenti pochi dati in quanto si è limitati ad un unico dispositivo. Immaginando, però, di installare tanti dispositivi di questo tipo su flotte di bici o monopattini elettrici, messi a disposizione da aziende di sharing, si potrebbe avere uno storico di dati per ciascun nodo e per ciascun momento della giornata. Questi dati potrebbero essere sfruttati dalle amministrazioni locali per individuare criticità e pianificare interventi che ridisegnino le città in modo da redistribuire e diminuire il traffico, quindi avere meno inquinamento.*

*La combinazione di questi dati, e del modello SARIMAX creato precedentemente, darebbe la possibilità di suggerire agli utenti che strada percorrere, dato l'orario ed i nodi di partenza e di arrivo. L'obiettivo ultimo di tutto ciò è quello di salvaguardare la salute dei cittadini.*

## 3.1 Componenti hardware

Il primo passo da intraprendere è selezionare i componenti che siano conformi con le specifiche del progetto e verificarne la compatibilità. L'elenco dei componenti selezionati è il seguente:

- Arduino UNO WiFi REV2;
- PMS7003/G7 sensore digitale PM2.5;
- Adafruit GPS PA1616S;
- Adafruit GPS external ective antenna;
- Breadboard kit;
- luci led;
- resistenze;

- scatola derivazione esterna IP65 ;
- accessoristica varia (collegamenti, pannello in plexiglass, elementi per fissaggio componenti, etc).

L'obiettivo è sfruttare la breadboard per connettere Arduino UNO alle seguenti componeti:

- GPS: fornirà i dati della posizione;
- sensore PMS7003: fornirà i dati di PM1, PM2.5 e PM10;
- led: segnaleranno la connessione del GPS ad almeno 4 satelliti e la corretta trasmissione dei dati, tramite Wi-Fi, al telefono.

Il GPS dovrà essere supportato dall'antenna esterna per aumentare la precisione. Le luci, invece, necessitano di apposite resistenze da 330 Ohm, collegate alla breadboard, per poter funzionare correttamente. Tutte le componenti dovranno essere inserite in una scatola di derivazione IP65, la quale, però, avrà come coperchio un pannello in plexiglass appositamente tagliato. Questo ultimo aspetto sarà fondamentale per poter vedere i led all'interno. Inoltre, si pone la necessità di eseguire dei fori in tale scatola per poter garantire un flusso d'aria ottimale all'interno della stessa e, di conseguenza, un corretto rilevamento dei valori di PM10. Infine, i dati inviati al telefono tramite Wi-Fi saranno processati da Node-RED e salvati sul DB Apache CouchDB.

Ora analizziamo nel dettaglio le specifiche di ciascuna componente per poter comprendere le scelte progettuali effettuate.

La prima componente che analizziamo è, sicuramente, Arduino UNO WiFi Rev.2, cuore dell'intero dispositivo. Arduino UNO WiFi Rev.2 fa parte della famiglia UNO dei dispositivi Arduino e viene ampiamente usato in tutti i progetti IoT che hanno come obiettivo la costruzione di una rete di sensori connessa al Wi-Fi o ad un dispositivo Bluetooth. La selezione di questo dispositivo, rispetto alle controparti, è motivata dalla possibilità di connetterlo ad una rete Wi-Fi utilizzando il suo acceleratore di chip crittografico ECC608 che ne garantisce la sicurezza. Esso incorpora un microcontrollore a 8 bit ATmega4809 di Microchip e dispone di una IMU (Inertial Measurement Unit) LSM6DS3TR integrata. Il modulo Wi-Fi è un SoC che permette l'accesso ad una rete Wi-Fi o può fungere da punto di accesso. Il modulo in questione è il NINA-W102; nella Figura 3.1 sono riportate le specifiche.

Come si può osservare, gli standard coperti e le prestazioni Wi-Fi sono ampiamente al di sopra dei requisiti prefissati.

Tornando nuovamente a parlare di Arduino UNO WiFi Rev.2, possiamo dire che dispone di 14 pin di ingresso/uscita digitale (5 possono essere utilizzati come uscite PWM), 6 ingressi analogici, una connessione USB, un jack di alimentazione, un ICSP e un pulsante di ripristino. Come verrà spiegato successivamente, basta un collegamento tramite standard USB per essere alimentato. Si sceglierà, quindi, di usare una powerbank per fornire energia al dispositivo, considerando che sono necessari 6 - 20V in input e che esso opera a 5V.

Dalla Figura 3.2 è chiaro come, seppure sia un dispositivo consumer, il livello di complessità e miniaturizzazione delle componenti sia piuttosto elevato. Le dimensioni sono di soli 68.6 mm di lunghezza e 53.4 mm di larghezza ed il peso è di appena 25g. Queste ultime caratteristiche sono di fondamentale importanza perché, dovendo installare il dispositivo finale su una bici o su un monopattino, bisogna rendere il tutto il più discreto e meno ingombrante possibile.

Il secondo dispositivo che analizziamo è il sensore per il recupero dei valori di PM10. La famiglia di sensori PMS7003 è in grado di raccogliere dati di PM1, PM2.5 e PM10, il che garantisce maggiore flessibilità e la possibilità di miglioramenti futuri del progetto. Stiamo

| | NINA-W101 | NINA-W102 |
|---|---|---|
| **Grade** | | |
| Automotive | | |
| Professional | • | • |
| Standard | | |
| **Radio** | | |
| Bluetooth qualification | v4.2 | |
| Bluetooth profiles | S, G, A2, AV, HF, HS | |
| Bluetooth BR/EDR | • | • |
| Bluetooth low energy | • | • |
| Bluetooth output power EIRP [dBm] | 8 | 8 |
| Wi-Fi 2.4 / 5 [GHz] | 2.4 | 2.4 |
| Wi-Fi IEEE 802.11 standards | b/g/n | b/g/n |
| Wi-Fi output power EIRP [dBm] | 18 | 18 |
| Max Wi-Fi range [meters] | 500 | 400 |
| Antenna type | p | i |
| **Application software** | | |
| Open CPU for embedded customer applications | • | • |
| **Interfaces** | | |
| UART | ♦ | ♦ |
| SPI | ♦ | ♦ |
| I²C | ♦ | ♦ |
| I²S | ♦ | ♦ |
| RMII | ♦ | ♦ |
| GPIO pins | 20 | 20 |
| AD converters (ADC) | ♦ | ♦ |
| **Features** | | |
| Point-to-Point Protocol | ♦ | ♦ |
| Extended Data Mode | ♦ | ♦ |
| Low Energy Serial Port Service | ♦ | ♦ |
| Wi-Fi throughput [Mbit/s] | 100 | 100 |
| Maximum Bluetooth connections | 8 | 8 |
| Micro Access Point [max stations] | 4 | 4 |
| Wi-Fi enterprise security | ♦ | ♦ |
| Secure boot | ♦ | ♦ |
| WPA/WPA2 | ♦ | ♦ |

**Figura 3.1:** Specifiche del modulo Wi-Fi NINA-W102

parlando di un sensore laser in grado di misurare il numero particelle in sospensione in un volume unitario di aria. Questo viene fatto utilizzando la teoria della dispersione della luce, cioè la dispersione della radiazione laser nelle particelle sospese nell'aria, e rilevando la luce diffusa con un angolo specifico, al fine di ottenere l'intensità di diffusione rispetto ad una curva temporale. I dati misurati vengono, così, raccolti ed elaborati da un microprocessore che, mediante trasformate di Fourier ed una serie di algoritmi complessi, ottiene il numero di particelle di una determinata dimensione. Di seguito alcune specifiche:

- *Tensione di funzionamento: 4.95 ̃5.05Vcc*
- *Corrente massima: 120mA*
- *Misurazione particolato: 0.3 ̃0, 1.0 ̃2.5, 2.5 10 (um)*
- *Campo di misura pm: 0 ̃500 ug/m3*
- *Corrente standby:* $\leq$ *200uA*

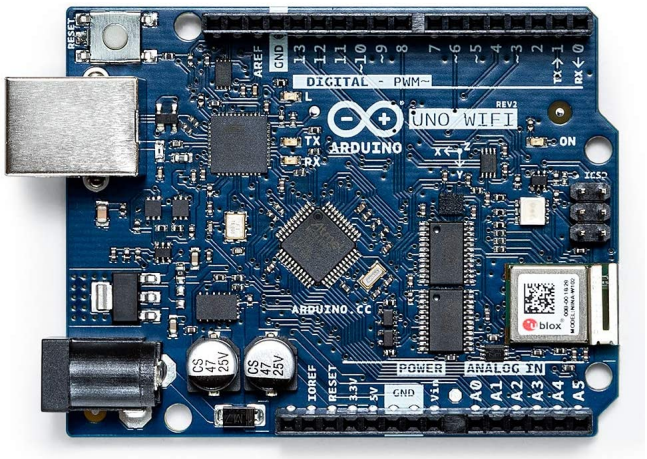


**Figura 3.2:** Immagine di Arduino UNO WiFi Rev.2

- *Tempo di risposta:* $\leq$ *10 s*
- *Temperatura di esercizio: -20 ̃50°C*
- *Intervallo umidità: 0 ̃99% RH*
- *Ripple della tensione inferiore a 100mV*
- *Tensione di alimentazione: 4.95 ̃5.05Vcc*
- *Potenza necessaria: 1W (5Vcc 200mA)*

Le dimesioni di soli 48 x 37 x 12 mm rientrano, ancora una volta, tra le principali motivazioni per cui si è scelto questo prodotto, che si può osservare in Figura 3.3. Come si vede dalla figura, con il prodotto vengono forniti anche cavi ed adattatori che permettono il collegamento con Arduino.


**Figura 3.3:** Immagine del sensore PMS7003 per il rilevamento della PM10

Il terzo dispositivo che è stato impiegato è Adafruit Ultimate GPS Breakout ed è osservabile in Figura 3.4. È costruito attorno al chipset MTK3339, un modulo GPS in grado di tracciare fino a 22 satelliti su 66 canali, ed ha una sensibilità di -165 dBm di tracciamento. Può eseguire fino a 10 aggiornamenti di posizione al secondo per la registrazione o il tracciamento.

Il consumo energetico è di 20 mA durante la navigazione. Il LED sul dispositivo lampeggia a circa 1 Hz durante la ricerca dei satelliti e una volta ogni 15 secondi quando viene trovata una correzione.

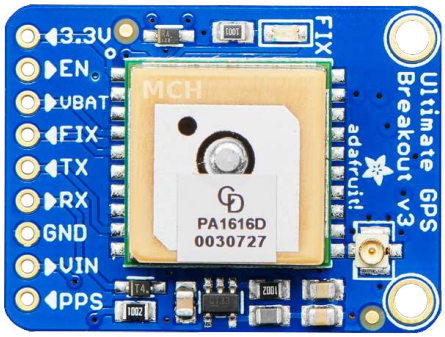


**Figura 3.4:** Immagine di Adafruit Ultimate GPS Breakout

Il modulo GPS ha un'antenna in ceramica standard che, come detto, permette ad esso di avere una sensibilità di -165 dBm. Quando, però, si desidera avere un'antenna più grande, è possibile agganciare qualsiasi antenna GPS attiva da 3 V tramite il connettore uFL. Quest'ultima funzionalità è stata sfruttata collegando un'antenna esterna che assorbe circa 10 mA e permette di ottenere 28 dB di guadagno. Unica nota negativa sull'impiego di questa antenna è la lunghezza di 5 metri del cavo che va in contrasto con l'obiettivo di ottenere un dispositivo finale di dimensioni limitate. Nonostante ciò, l'antenna è stata fondamentale per limitare le tempistiche di aggancio del GPS ai satelliti e migliorare la precisione. Si è passati da tempi iniziali di tracciamento superiori a 10 minuti a tempi mai superiori a 1-2 minuti. Una volta ottenute le due componenti è stato necessario collegarle tra loro tramite un cavo coassiale di 1,13 mm da standard U.fl IPX femmina a SMA femmina. Il risultato è presentato in Figura 3.5.

Come visto, le componenti precedenti sono di alta qualità ed adatte all'impiego nel progetto. È ora necessario comprendere come queste siano collegate tra loro, mediante la breadboard, nelle Figure 3.6 e 3.7 è possibile avere lo schema dei collegamenti (la componente in alto a sinistra rappresenta il sensore di PM10), realizzato in Frizzing, ed un focus reale che mette in risalto alcuni dettagli.

Una volta che tutte le componenti sono collegate tra loro, esse vengono fissate alla breadboard con del velcro. L'impiego di questa soluzione è motivato sia dalla possibilità di rimuovere facilmente ciascuna componente, in caso di problemi, che dalla stabilità e dall'attenuazione agli urti che esso conferisce.

Prima di porre la breadboard all'interno della scatola sono stati eseguiti degli interventi di modifica su quest'ultima. Il primo intervento consiste nella creazione di un foro che coincide con l'ingresso del cavo Usb Tipo B per il collegamento tra powerbank e scheda Arduino. Un secondo intervento ha l'obiettivo di creare 10 piccoli fori per ciascun lato corto della scatola, disposti su due righe e distanti poco più di un centimetro l'uno dall'altro. Questi hanno lo scopo di favorire il ricambio dell'aria all'interno della scatola in modo da rendere le misurazioni di PM10 corrette. L'ultima modifica apportata alla scatola riguarda la sostituzione

**Figura 3.5:** Immagine del collegamento tra il GPS e l'antenna esterna

del coperchio originale, completamente nero, con un pannello in plexiglass di dimensioni analoghe. In questo modo, oltre a dare un look premium al dispositivo, si risolve il problema di non poter vedere i led che segnalano quando il GPS riesce ad agganciare almeno 4 satelliti e quando la comunicazione Wi-Fi con il telefono è avvenuta con successo ed effettivamente i dati transitano tra i due dispositivi.

A questo punto la breadboard è stata riposta sul fondo della scatola e fissata con il velcro, per le stesse ragioni spiegate precedentemente. L'antenna è stata posizionata a lato della breadboard, nonostante la lunghezza di ben 5 m, ed è stata fissata anch'essa con del velcro alla scatola. Il risultato è un dispositivo piuttosto compatto dalle dimensioni di 20 x 12 x 6 cm. I risultati finali sono riportati nella Figura 3.8 e 3.9.

Nella prima immagine, si possono visualizzare tutte le componenti ed i collegamenti svolti.

Nella seconda, invece, si può osservare il risultato finale. Nel dettaglio, sono visibili i fori per il passaggio dell'aria e per il collegamento tramite USB, oltre alla copertura in plexiglass.

## 3.2 Codice implementativo

L'obiettivo di questa sezione è spiegare le operazioni più importanti che vengono svolte dal codice implementato su Arduino, al fine di ottenere i dati sulla posizione e sul PM10. Il codice implementato è stato sviluppato con Arduino IDE 2.1.0, Integrated Development Environment multipiattaforma basata su Electron, ed è un file con estensione `.ino` che verrà caricato sul microcontrollore. Esso si divide in due parti: fase di setup e fase di loop.

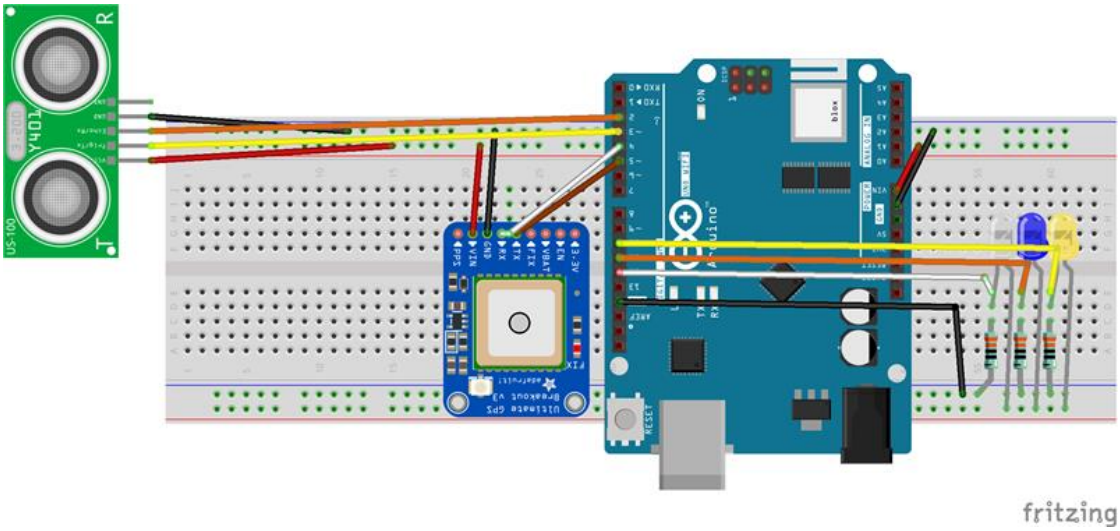


**Figura 3.6:** Schema dei collegamenti tra le componenti

### 3.2.1 Fase di setup

La prima parte del codice costituisce la fase di setup, eseguita allo start-up del componente, ed è caratterizzata dalle seguenti operazioni:

- Viene verificata la connettività al Wi-Fi (connessione avvenuta in seguito alla definizione del nome della rete Wi-Fi del telefono e della relativa password).
- Viene stabilita la connessione con il Broker MQTT.
- Viene mandato un messaggio di reset (alla ricezione di tale messaggio, il backend su Nodered stacca un nuovo codice univoco per il trip).
- Vengono spenti tutti i led in modo da ripartire da una situazione standard.

È possibile osservare lo schema di tale processo nella Figura 3.10.

### 3.2.2 Fase di loop

La seconda parte di codice costituisce una fase caratterizzata da un loop. I microcontrollori, infatti, eseguono all'infinito il codice caricato sulla loro scheda generando, quindi, un loop infinito.

La frequenza a cui questo loop opera su Arduino UNO è 5000 millisecondi. È stato deciso di abbassare l'intervallo di campionamento per:

- Evitare un uso eccessivo della batteria
- Evitare di collezionare dati molto simili, presi a distanza di pochi millisecondi l'uno dall'altro, che non apporterebbero vantaggi significativi ed allungherebbero i tempi di elaborazione delle operazioni successive

Ad ogni loop viene controllato se il tempo che intercorre tra l'ultimo valore registrato e il tempo attuale è maggiore di questo intervallo. In caso affermativo si procede ad un nuovo campionamento, in particolare:

- viene letto il valore del sensore PM10;

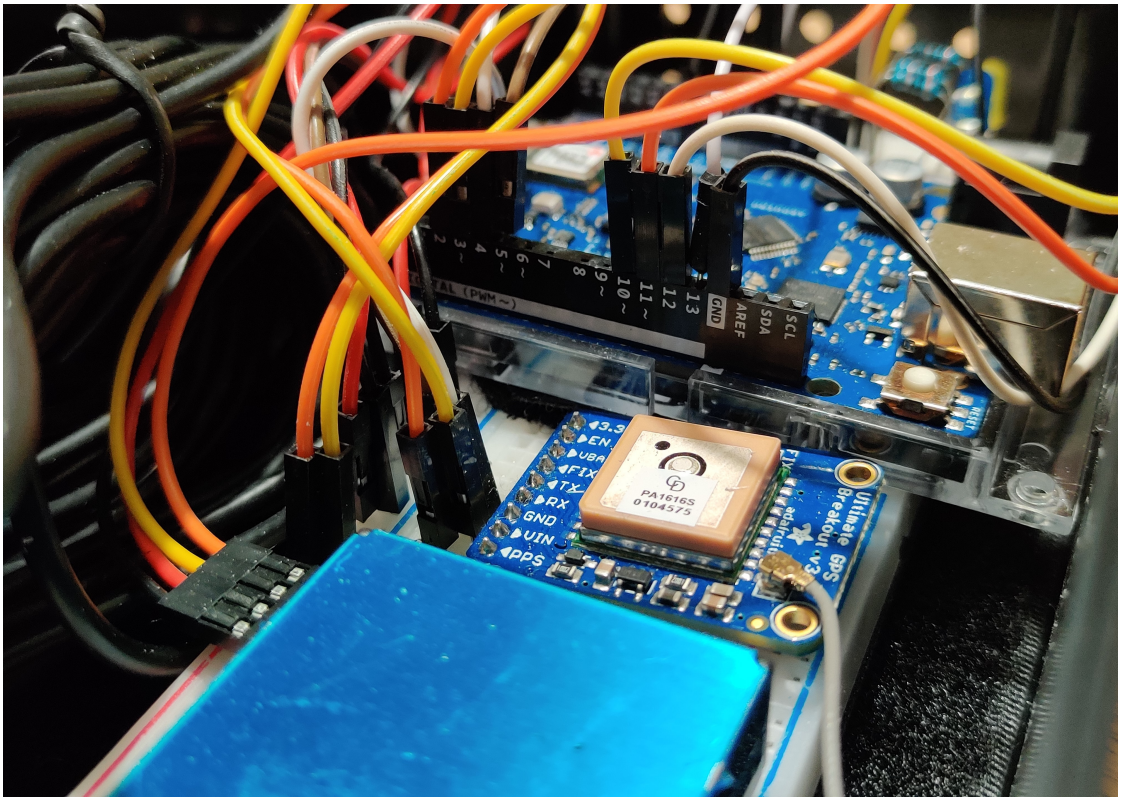


**Figura 3.7:** Foto con focus su alcune componenti e collegamenti

- viene letto il valore del sensore GPS.

Si ricorda che è necessario indicare, precedentemente, i pin dal quale Arduino può gestire e scambiare dati con i sensori: pin 4 RX e 5 TX per il sensore GPS e pin 2 RX e 3 TX per il sensore di rilevamento del particolato. In Figura3.11 si può osservare lo schema di tale loop.

Se il valore del PM10 è valido (valore maggiore di 0) ed anche il valore del GPS è valido (almeno 4 satelliti agganciati e latitudine/longitudine popolate), il dato viene mandato al Broket MQTT sul topic “bike”.

### Lettura dati PM10

Il sensore PMS7003 legge i dati sul particolato e li invia, tramite porta seriale, come byte array. Nel data sheet del componente (`https://www.espruino.com/datasheets/PMS7003.pdf`), a pagina 13, si può vedere il valore che ogni cella di byte rappresenta. Sebbene tutte le tipologie di PM calcolate vengono lette sia come $ug/m^3$ che come numero di particelle in 0.1L di aria, sono stati inviati verso il broker MQTT solo i dati di PM10 in $ug/m^3$. Nel datasheet questi dati corrispondono ai campi "Data 3 high 8 bits" e "Data 3 low 8 bits".

### Lettura dati GPS

Il sensore GPS Adafruit legge i dati e li decodifica come stringa, CSV, GPGGA. In particolare, si seleziona GPGGA da diverse frasi NMEA ovvero:

- $GPVTG - Indica il Track Made Good ed il valore di Groud Speed;
- $GPGGA - Contiene le informazioni di latitudine, longitudine, quota e tempo;

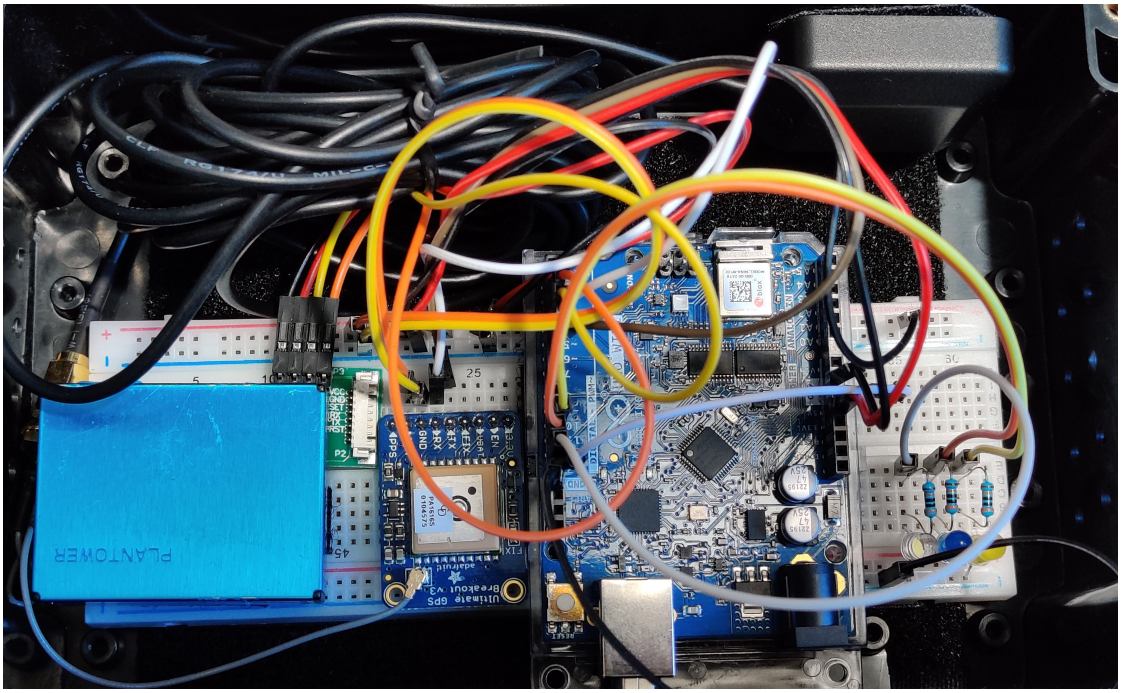

**Figura 3.8:** Foto del risultato finale in cui sono visibili tutti i collegamenti

- $GPGSA - Contiene dati relativi al DOP (Diluizione della precisione) e al PRN (pseudo random noise) dei satelliti utilizzati per il fix;
- $GPRMC - Indica il valore minimo raccomandato dei dati di tempo, velocità e posizione.

Identificando i valori separati dalla virgola, per GPGGA, abbiamo i campi visibili in Tabella 3.1.

| Campo | Struttura | Descrizione | Simbolo | Esempio |
|---|---|---|---|---|
| 1 | $GPGGA | Log header | | $GPGGA |
| 2 | utc | UTC time status of position | hhmmss.ss | 202134.00 |
| 3 | lat | Latitude (DDmm.mm) | llll.ll | 5106.9847 |
| 4 | lat dir | Latitude direction (N = North, S = South) | a | N |
| 5 | lon | Longitude (DDDmm.mm) | yyyyy.yy | 11402.2986 |
| 6 | lon dir | Longitude direction (E = East, W = West) | a | W |
| 7 | quality | refer to Table: GPS Quality Indicators | x | 1 |
| 8 | # sats | Number of satellites in use | xx | 10 |
| 9 | hdop | Horizontal dilution of precision | x.x | 1.0 |
| 10 | alt | Antenna altitude above/below mean sea level | x.x | 1062.22 |
| 11 | a-units | Units of antenna altitude (M = metres) | M | M |
| 12 | undulation | Relationship between geoid and WGS84 ellipsoid | x.x | -16.271 |
| 13 | u-units | Units of undulation (M = metres) | M | M |
| 14 | age | Age of correction data (in seconds), Max 99 sec | xx | [empty] |
| 15 | stn ID | Differential base station IDr | xxxx | [empty] |
| 16 | *xx | Checksum | *hh | *48 |
| 17 | [CR][LF] | Sentence terminator | | [CR][LF] |

**Tabella 3.1:** Dati inviati dal sensore GPS

Lato codice vengono individuati e propagati solo quelli utili all'identificazione della posizione (alcuni saranno comunque propagati per eseguire test vari), ovvero:

- Lat (campo 3);

**Figura 3.9:** Foto della scatola chiusa con pannello in plexiglass

- Lat dir (campo 4);
- Lon (campo 5);
- Lon dir (campo 6);
- #sats (campo 8).

## 3.3 Real Time Processing su NodeRed

Node-RED è uno strumento di programmazione per collegare dispositivi hardware, API e servizi online. Esso fornisce un editor basato su browser che semplifica il collegamento dei flussi utilizzando un'ampia gamma di nodi, da una palette di cui è possibile fare il deploy a runtime.

Tornando al progetto, una volta che i dati arrivano sul Broker MQTT, viene eseguito un flusso NodeRed che li consuma e li modifica, per prepararli all'inserimento nella base dati. In particolare:

- la latitudine e la longitudine, espresse in decimali, vengono convertite in radianti per un uso più agevole durante la fase di analisi dati;

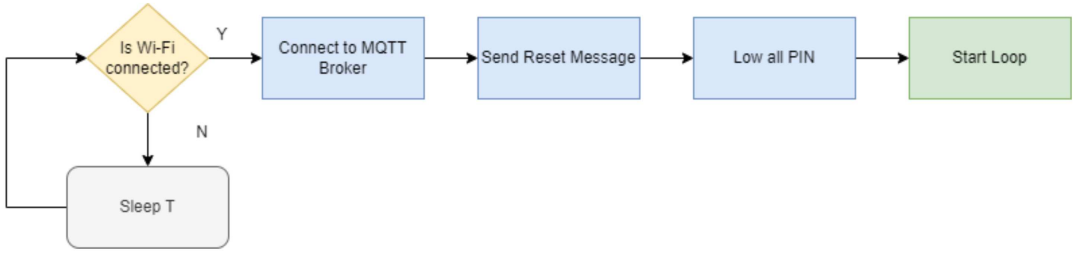


**Figura 3.10:** Schema della fase di setup del codice Arduino

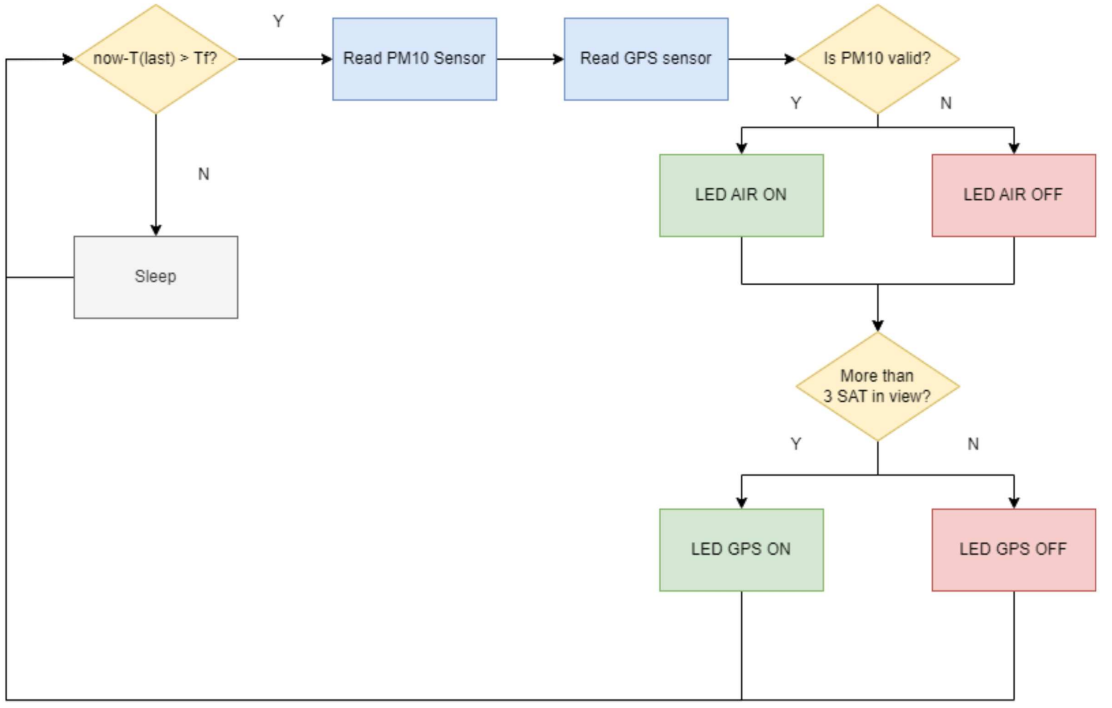


**Figura 3.11:** Schema della fase di loop del codice Arduino

- la direzione di latitudine e longitudine (campo 4 e 6 della Tabella 3.1), utili alla conversione da decimali a radianti, non vengono propagati verso gli step successivi;
- i dati numerici inviati da Arduino come stringhe vengono convertiti in double;
- viene aggiunto al payload un timestamp di ricezione;
- i dati vengono strutturati nel formato utile a Couch DB per l'inserimento.

Nella Figura 3.12 si possono osservare i due flussi che permettono di eseguire le operazioni citate precedentemente.

Una volta effettuata questa fase di pulizia e preparazione, Couch DB viene invocato tramite API Rest per inserire la riga nel database.

## 3.4 Raccolta dei dati

La raccolta dei dati consiste, semplicemente, nel fissare il dispositivo basato su Arduino alla bici, collegarlo ad una powerbank ed attendere che i led di connessione, ai satelliti e al

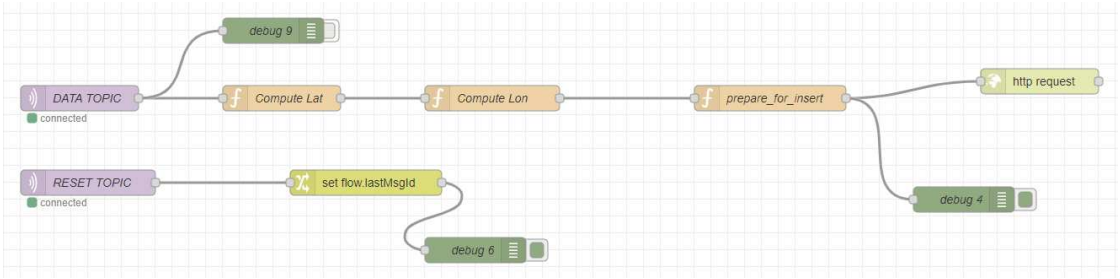


**Figura 3.12:** Flussi di elaborazione real time in Node-RED

Wif-Fi, si accendano. A questo punto si può fare un giro per il quartiere San Salvario e, al termine, si scollega la fonte di alimentazione per interrompere l'invio dei dati. Nel frattempo i dati sono stati trasferiti da Arduino a Nodered, che invia i dati alla base dati Coutch DB secondo le modalità spiegate precedentemente.

Apache CouchDB è un DBMS relazionale (NoSQL) nato nel 2005 ed entrato a far parte del mondo Apache nel 2008. Esso utilizza JSON per memorizzare i dati, JavaScript come linguaggio di interrogazione, e fa uso di MapReduce e HTTP come API.

Nella Tabella 3.2 vengono visualizzati i campi del database, con una breve descrizione ed un esempio.

| Campo | Descrizione | Esempio |
|---|---|---|
| _id | Identificativo della struttura dati | "5cc441bf521fbe69411c25b3f4c591fb" |
| _rev | - | "1-a1f52caa73ef2e03285ce5835bcec566" |
| pm100 | Valore di PM10 | 4 |
| pm25 | Valore di PM2.5 | 4 |
| s | Numero di satelliti (stringa) | "06" |
| lt | Latitudine in decimali | "4503.5415" |
| ln | Longitudine in decimali | "00740.8434" |
| d_lt | Direzione latitudine | "N" |
| d_ln | Direzione longitudine | "E" |
| q | Qualità | "1" |
| hd | Hdop | "1.29" |
| lat | Latitudine in gradi | 45.059025 |
| lon | Longitudine in gradi | 7.680723333333333 |
| satellites | Numero di satelliti | 6 |
| ts | Timestamp rilevazione | 1682698933469 |
| tripId | Identificativo del trip | "TRIP_1682698931799" |

**Tabella 3.2:** Struttura del database CouchDB

Questo costituirà il punto di partenza grazie al quale verrà popolato il grafo NetworkX, con i dati acquisiti da Arduino, e verrà costruito un dataset per ciascun nodo, con i relativi dati.

CAPITOLO 4

# Creazione grafo della rete stradale

*L'obiettivo di questo capitolo è creare un grafo NetworkX che permetta di associare a ciascun nodo i valori raccolti sia dalle stazioni fisse che dal dispositivo basato su Arduino. L'associazione dei dati, provenienti da Arduino, con il nodo della mappa più vicino viene fatta grazie ad un algoritmo che confronta le coordinate provenienti dal GPS del dispositivo con quelle associate a ciascun nodo della mappa stessa. La creazione del grafo e l'associazione dei vari attributi a ciascun nodo saranno propedeutici alla costruzione dell'algoritmo che calcolerà il percorso migliore dal punto di vista salutare.*

## 4.1 Grafo NetworkX

In questa sezione si vuole creare un grafo NetworkX delle strade di San Salvario, quartiere di Torino .

NetworkX è una libreria Python, con licenza BSD-new, costruita con lo scopo di studiare grafi e reti. La ragione principale per la quale è stato deciso di impiegare tale libreria sta nelle feature che mette a disposizione, queste sono:

- presenza di classi per grafici e digrammi;
- conversione di grafici da e verso diversi formati;
- capacità di costruire grafici casuali o di costruirli in modo incrementale;
- possibilità di trovare sottografi, clique e k-core;
- possibilità di esplorare adiacenza, grado, diametro, raggio, centro, mezzo, etc.;
- capacità di disegnare reti in 2D e 3D.

Nel caso specifico ciascun nodo definisce l'intersezione di più strade, o la fine di una strada chiusa, ed è caratterizzato da un identificatore, valore numerico incrementale, e da un attributo *pos*, che definisce la latitudine e la longitudine del nodo stesso.

Un esempio di creazione di un nodo è il seguente:

```
G.add_node(0, pos = (45.026065, 7.668954))
```

Una volta creati i nodi della mappa è necessario definire gli archi che corrispondono alle strade. In questo caso la definizione di un arco è piuttosto semplice in quanto basta indicare i nodi nei quali esso sussiste. Un esempio di creazione di un arco è il seguente:

```
G.add_edge(0,1)
```

A questo punto il grafo è completo e viene stampato sfruttando la funzione *show()* della libreria matplotlib. Il risultato è visibile in Figura 4.1.

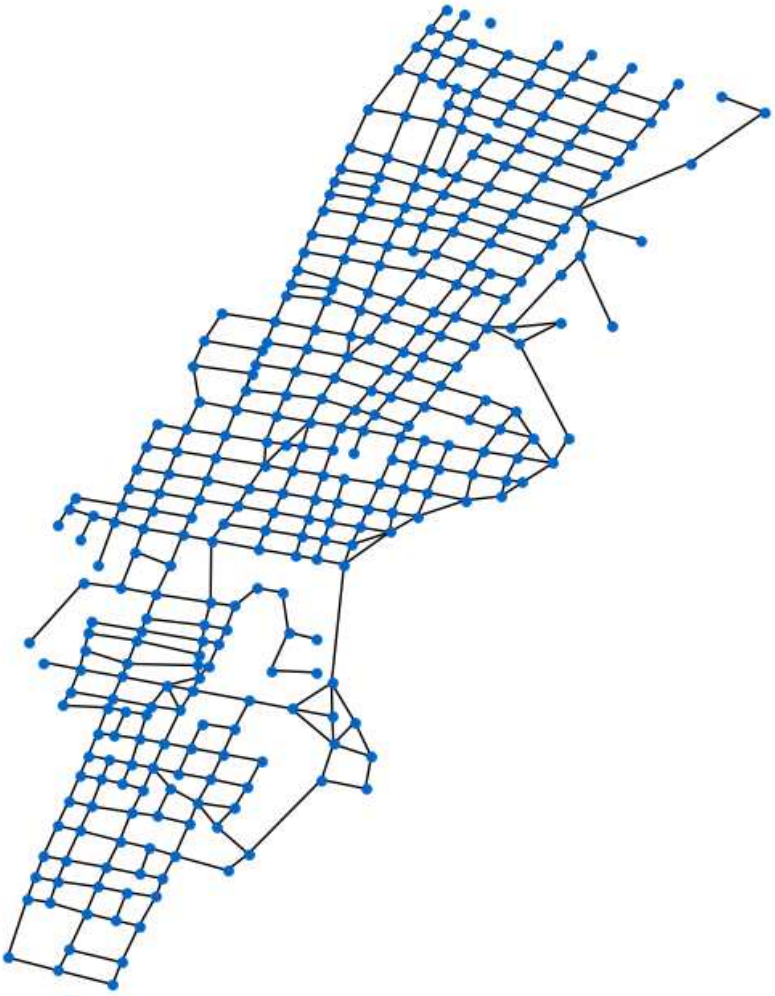

**Figura 4.1:** Grafo creato su NetworkX per il quartiere San Salvario di Torino

Dalla figura precedentemente citata si riconosce la disposizione delle strade a scacchiera, caratteristica della città di Torino, ad eccezione di qualche area nella parte a destra dell'immagine. Queste aree corrispondono al Parco del Valentino, che si affaccia sul fiume Po, e, rappresentando uno dei principali polmoni verdi della città, dovrebbe essere caratterizzato da una qualità dell'aria migliore.

Per poter salvare il grafo è stato sfruttato il modulo pickle che permette di implementare protocolli binari per serializzare e deserializzare una struttura di oggetti Python. "Pickling" è il processo mediante il quale una gerarchia di oggetti Python viene convertita in un flusso di byte e "unpickling" è l'operazione inversa (sono l'equivalente di serialization o marshalling). Il file ottenuto è nominato *mapTurin.pickle*.

## 4.2 Analisi del grafo

L'analisi del grafo, che rappresenta le strade di Torino, non è una operazione necessaria ai fini della tesi ma può rappresentare uno spunto importante per comprendere quali siano i

nodi centrali ed in cui si potrebbe concentrare l'afflusso di auto, con possibili valori alti di PM10. Inoltre, analisi di questo tipo permettono di verificare la correttezza del grafo creato.

In Figura 4.2 è riportato il risultato del primo check che eseguiamo; esso riguarda il numero di nodi e di archi che compongono il grafo, ovvero 345 nodi e 587 archi.

```
Graph with 345 nodes and 587 edges
```

**Figura 4.2:** Numero di nodi ed archi che compongono il grafo

Una funzione interessante, messa a disposizione dalla libreria, è *diameter()*. Definiamo diametro la massima eccentricità del grafo. L'eccentricità di un nodo, a sua volta, è la distanza massima del nodo stesso da tutti gli altri nodi del grafo. Quanto restituito da questa funzione è interessante in quanto ci da una misura dell'ampiezza del quartiere, più precisamente del numero massimo di intersezioni e di strade, che si possono incontrare andando da una parte all'altra dello stesso. Come si può notare dalla Figura 4.3 il numero è piuttosto alto, anche a causa della geometria allungata.

```
Network diameter:  39
```

**Figura 4.3:** Diametro del grafo

Viene, quindi, verificato se il grafo è connesso e vengono calcolati il numero di componenti connesse che lo costituiscono ed il numero di nodi o archi da rimuovere per rendere il grafo disconnesso. La definizione della teoria dei grafi dice che un grafo G = (V, E) è detto connesso se, per ogni coppia di vertici $(u, v) \in V$, esiste un cammino che collega $u$ a $v$. La verifica di tale condizione ci garantisce che le strade della mappa creata permettono di arrivare in tutti i punti prefissati. Ciò che si ottiene è riportato in Figura 4.4.

```
Graph is connected: True
Number of different connected components:  1
Number of nodes to be removed (to be disconnected):  1
Number of edges to be removed (to be disconnected):  1
```

**Figura 4.4:** Verifica della connessione del grafo

Il risultato True conferma la corretta costruzione del grafo. Il numero di nodi/archi da rimuovere pari ad uno ci dice, banalmente, che esistono delle strade senza uscita.

Un altro parametro interessante, usato spesso nelle analisi dei grafi, è il raggio; esso rappresenta l'eccentricità minima. Di conseguenza vengono individuati i nodi che rappresentano il centro del grafo e, quindi, del quartiere. Queste zone potrebbero rappresentare gli snodi principali del traffico, con possibili valori alti di PM10. La chiusura delle triadi fornisce una indicazione di come sono connessi tra loro i nodi. Nella teoria dei grafi la triade rappresenta una struttura a tre nodi tutti collegati tra loro; nel caso specifico, può essere utile per comprendere se da ciascun nodo possono essere intraprese strade alternative o se, alla chiusura di una strada, si possono presentare problemi per raggiungere un altro nodo. Infine, in Figura 4.5, si possono vedere anche quali sono i nodi che formano la cosiddetta periferia, cioè quei nodi con eccentricità pari al diametro del grafo.

I nodi individuati sia per il centro del grafo che per la periferia sono conformi a quanto ci si aspetta. Il coefficiente di chiusura delle triadi non è molto alto; ciò è dovuto anche alla struttura a scacchiera della mappa di Torino.

```
Radius (minimum eccentricity): 20
Center:  [173, 174, 175]
Triadic closure: 0.03893575600259572
Periphery:  [2, 9, 344]
```

**Figura 4.5:** Raggio, centro, chiusura delle triadi e periferia del grafo

In Figura 4.6 vengono visualizzati i 10 nodi con centralità maggiore, presupponendo che la centralità per un nodo $v$ è la frazione di nodi a cui è connesso.

```
Top 10 centrality nodes:
[(34, 0.014577259475218658), (43, 0.014577259475218658), (50, 0.014577259475218658), (
```

**Figura 4.6:** I dieci nodi con i valori più alti di centralità

A questo punto si esegue una prova utile solo a capire cosa ci attenderà nei capitoli successivi. Infatti, l'obiettivo finale sarà quello di proporre all'utente il cammino migliore sulla base dell'inquinamento di ciascuna strada. In particolare, si vuole eseguire un piccolo test in cui viene indicato il percorso più rapido (presupponendo che il peso su ciascun arco sia unitario) fra i tre nodi con centralità più alta.

```
SHORTEST PATH BETWEEN TOP 3 NODES BY CENTRALITY:
Path34 - 43:  [34, 35, 43]
Path34 - 50:  [34, 32, 31, 47, 50]
Path43 - 34:  [43, 35, 34]
Path43 - 50:  [43, 35, 34, 32, 31, 47, 50]
Path50 - 34:  [50, 47, 31, 32, 34]
Path50 - 43:  [50, 47, 31, 32, 34, 35, 43]
```

**Figura 4.7:** Cammino più corto tra i migliori tre nodi per centralità

Dalla Figura 4.7 si può notare che i nodi con centralità più alta, spesso ma non sempre, sono relativamente vicini.

Viene stampato anche il coefficiente di clustering medio per il grafo (Figura 4.8). Questo parametro si ottiene ripetendo $n$ volte le seguenti operazioni: si sceglie un nodo a caso, si scelgono poi due dei suoi vicini e si controlla se sono connessi. Il numero basso è giustificato anche questa volta dalla struttura della città.

La distribuzione dei coefficienti di centralità dei nodi è mostrata nella Figura 4.9 e mostra un picco nell'intorno di 0.012 e di 0.009, quindi pochi nodi hanno valori di centralità che si discostano da questi valori.

Vediamo la stessa cosa, ma con una visualizzazione basata sulla posizione geografica dei nodi, in Figura 4.10. Questa visualizzazione permette di capire, grazie ai colori associati ai vari nodi, come si distribuisce la centralità. Abbiamo, ovviamente, che le strade a centralità più alta si trovano nelle zone centrali del quartiere. In alcuni casi è vero che queste strade sono le più trafficate, quindi potrebbero avere livelli di PM10 più alti. Non sempre, però, questa regola è verificata in toto. Ad esempio, via Nizza, che si snoda da Nord a Sud nella parte ovest della mappa ed ha nodi in arancione, è una via molto trafficata in quanto connette il centro con aree di grande importanza, come il Lingotto o il grattacielo della Regione Piemonte.

```
Avarage clustering coefficient:
0.03217054263565891
```

**Figura 4.8:** Coefficiente di clustering medio

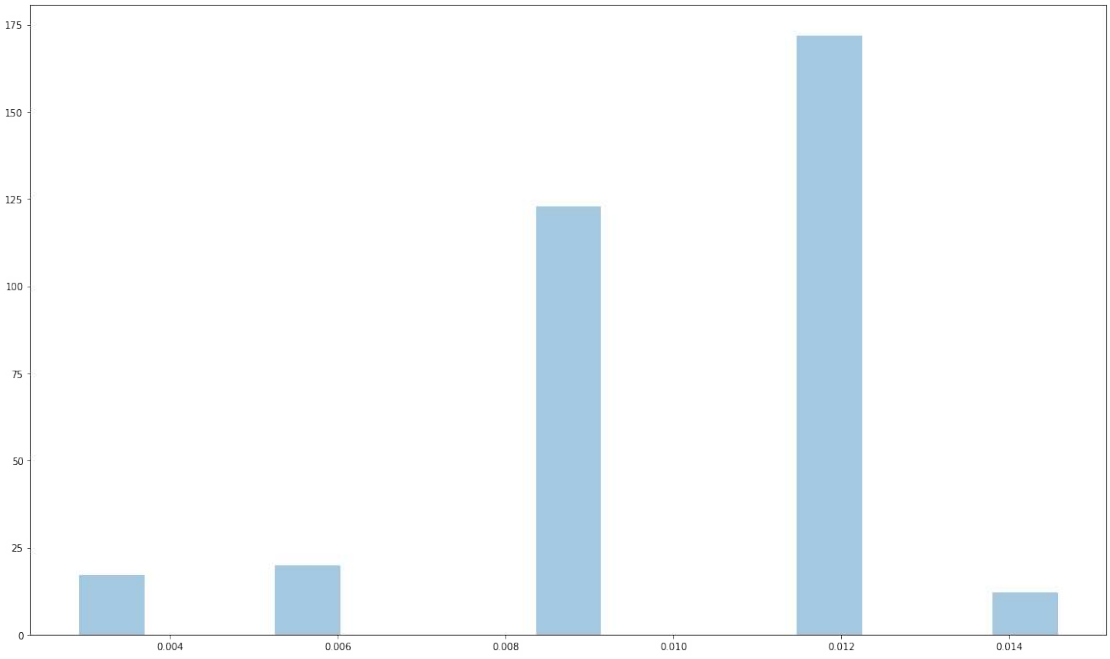


**Figura 4.9:** Distribuzione dei coefficienti di centralità dei nodi

## 4.3 Associazione valori stazioni fisse

Lo scopo di questa sezione, e della successiva, è associare i dati a disposizione sulla PM10, ricavati sia da stazioni fisse che da Arduino, a ciascun nodo della mappa creata precedentemente. Questo permetterà di sviluppare servizi che diano la possibilità di restituire valori di PM10 di un nodo specifico, passati o futuri, di calcolare il percorso con meno particolato e di visualizzare una heatmap di San Salvario. In particolare, in questa sezione, si vogliono salvare i dati di PM10 provenienti dalle stazioni fisse. L'idea iniziale prevedeva il recupero dei dati provenienti dalle tre stazioni fisse ed il calcolo, per ciascun nodo, del valore, eseguendo una interpolazione basata sulla posizione geografica del nodo stesso. Questo processo prevede innanzitutto il salvataggio delle coordinate delle tre stazioni ovvero:

- Torino Consolata = 45.075694, 7.678006
- Torino Lingotto = 45.024883, 7.648990
- Torino Rubino = 45.042375, 7.625804

La funzione chiave prevede prima una conversione delle coordinate da $lat/lon$ a $x/y$ e poi il calcolo del valore interpolato. Per quanto concerne la conversione si passa da un EPSG 4326 ad un EPSG 7131, ricordando che EPSG è un registro pubblico di dati geodetici, sistemi di riferimento spaziali, ellissoidi terrestri, trasformazioni di coordinate e relative unità di misura. Seppur parzialmente sviluppata, questa idea è stata man mano abbandonata per varie ragioni. Innanzitutto l'interpolazione ha senso se i dati delle stazioni differiscono abbastanza tra loro e la dimensione della mappa è comparabile con l'area coperta dal triangolo che ha come vertici le tre stazioni. Purtroppo, queste condizioni non si verificano in toto ed, in particolare, la

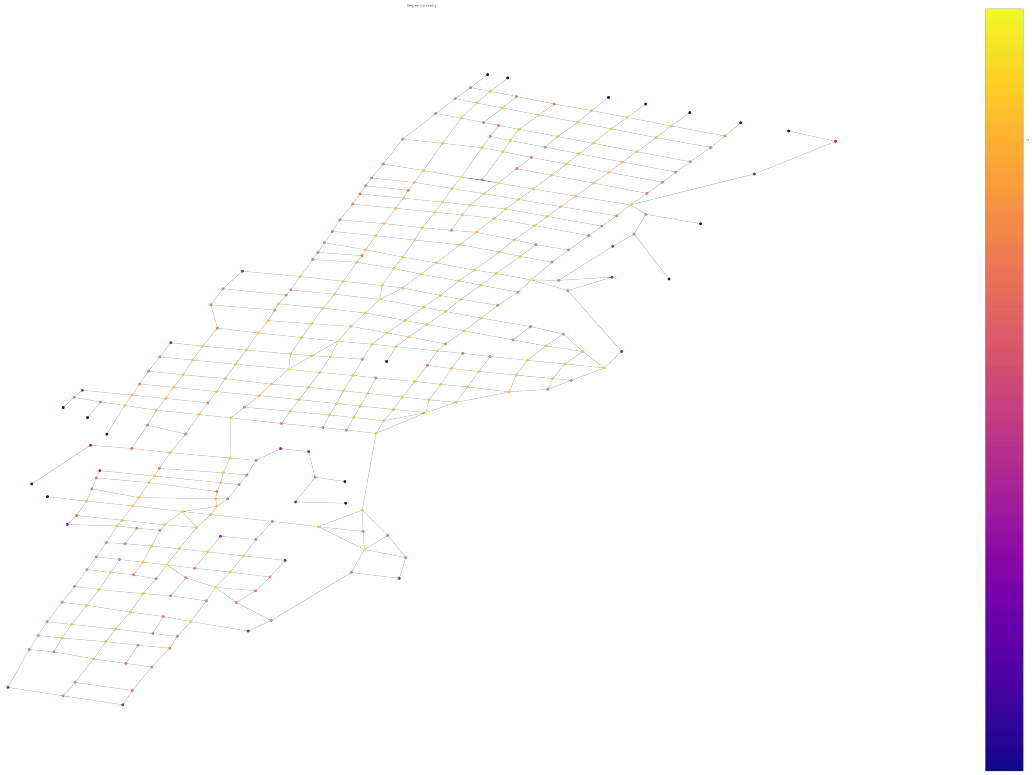

**Figura 4.10:** Mappa dei coefficienti di centralità dei nodi

mappa presa in esame è del solo quartiere San Salvario, quindi l'interpolazione non produce risultati tanto diversi tra i vari nodi. La ragione per cui è stato deciso di lasciare questo sottocapitolo, ed il relativo codice sviluppato, è che in futuro si potrebbe espandere la mappa all'intera città di Torino e quindi poter apprezzare il valore apportato da queste operazioni. Inoltre, in futuro potrebbero essere installate ulteriori stazioni fisse che renderebbero i valori interpolati ancora più precisi e sensati. Si è deciso di usare i dati di PM10 calcolati come mediana dei valori provenienti dalle tre stazioni fisse. Questi vengono salvati in un dizionario che ha come chiavi le date relative ai vari giorni e come valori la PM10. Per ciascun nodo si salva tale dizionario come attributo "stazioniFisse". Le motivazioni di tale associazione risiedono sia nella volontà di rendere ciascun nodo indipendente, possedendo tutti i dati utili, che nella possibile evoluzione citata precedentemente.

## 4.4 Associazione valori puntuali e costruzione dei dataset

In questa sezione si vuole scrivere del codice che abbia come obiettivo finale la creazione di dataset, costruiti a partire dai dati acquisiti da Arduino, da sfruttare nei servizi messi a disposizione per gli utenti. Si ricorda che tali servizi riguarderanno la restituzione dei valori di PM10, dati il nodo e la data di interesse, la restituzione del percorso meno inquinato dati i nodi di partenza e di arrivo, ed, infine, la visualizzazione di una heatmap. Per poter perseguire questo obiettivo si dovranno eseguire varie operazioni, ovvero:

- connessione a Coutch DB per il recupero dei dati;
- associazione dei valori recuperati dal DB con i nodi vicino ai quali si è transitato con la bici;

- calcolo del valore mediano fra tutti i valori di un viaggio che vengono associati ad un determinato nodo;
- estensione di tali valori a tutti i nodi della mappa (utile per i nodi in cui non si è passati direttamente);
- aggiunta di un attributo a ciascun nodo della mappa contenente tutti i valori storici associati a quel nodo;
- creazione dell'attributo "arduinoData", per ciascun nodo, che riassume, per ciascun giorno, a partire dal 1° aprile 2023, il valore di PM10 medio giornaliero ed il valore di PM10 medio per diverse fasce orarie.

La prima operazione, dunque, riguarda la connessione al database, che avviene tramite una chiamata POST con url http://172.16.8.196:5984/air_quality_sensor/_find , con un campo json pari a "selector": "_id": "$gt": None , "limit" : 100000 e con un header uguale a "Content-Type":"application/json". Il campo json ci permette di eliminare la possibilità di valori errati nel DB, in cui l'id è None, e di modificare il numero limite di elementi che vengono ritornati, che di default è pari a 25. La response viene letta come json e viene estratto il contenuto del campo docs:

*[{'_id': '5cc441bf521fbe69411c25b3f4c591fb', '_rev': '1-a1f52caa73ef2e03285ce5835bcec566', 'pm100': 4, 'pm25': 4, 's': '06', 'lt': '4503.5415', 'ln': '00740.8434', 'd_lt': 'N', 'd_ln': 'E', 'q': '1', 'hd': '1.29', 'lat': 45.059025, 'lon': 7.680723333333333, 'satellites': 6, 'ts': 1682698933469, 'tripId': 'TRIP_1682698931799'}, { ... } ... ]*

Adesso che si hanno tutti i valori del database salvati in una variabile, si vuole associare tali valori ai nodi vicino ai quali si è transitato con la bici. In questo modo si ripercorre lo spostamento eseguito in ciascun viaggio con la bici. Per fare ciò si importa il grafo e, per ciascun dato raccolto, viene calcolata la distanza con ciascun nodo sulla mappa. Per il calcolo della distanza è stata creata una funzione che riceve in input i valori di latitudine e longitudine sia della posizione in cui è stato raccolto il dato con Arduino che del nodo con il quale si vuole eseguire il confronto. Il risultato è la creazione, per ciascuna riga del database di partenza, di una lista in cui ogni elemento è un array che ha come primo elemento il nome del nodo della mappa e come secondo parametro la distanza con il punto di rilevamento. Fra gli elementi della lista si estrae quello che ha il secondo valore dell'array, quindi la distanza minore. Questa operazione viene fatta per ogni rilevazione e, quindi, vengono di volta in volta aggiunti degli elementi ad un dizionario che ha come chiavi il tripId e come valori degli array che portano con sè alcune informazioni utili come: timestamp di rilevamento, valore di PM10 acquisito, id del nodo più vicino al relativo rilevamento e distanza da esso. Il risultato di tale operazione è il seguente:

*{'TRIP_1682698931799': [[1682698933469, 4, 316, 0.03], [1682698939355, 4, 316, 0.03], [1682698944484, 5, 316, 0.03], [1682698950638, 5, 315, 0.04], [1682698955340, 5, 311, 0.04], [1682698960455, 6, 311, 0.04], [1682698965580, 6, 311, 0.04], ... ], 'TRIP_1682699679064': [[1682699882384, 7, 311, 0.04], [1682699887388, 6, 311, 0.04], [1682699892513, 7, 311, 0.04], [1682699898444, 8, 311, 0.04], ... ] ... }*

A questo punto si vuole risolvere un problema; infatti, se, ad esempio, l'utente si ferma o procede lentamente si avranno tantissimi valori associati ad un nodo che, di fatto, sono superflui. Inoltre, di tutti questi valori ci potrebbero essere degli outlier che falserebbero la validità della misurazione; pensiamo sia ad errori nella misurazione da parte del sensore che a situazioni in cui, per varie ragioni, ci sono punti in cui il valore di PM10 diventa particolarmente alto (ad esempio un auto molto inquinante parte molto vicino all'utente). La soluzione che è stata trovata prende in esame, di volta in volta, tutti i valori con un certo TripId associati ad un nodo specifico e ne calcola il valore mediano. Quindi, per ciascun

viaggio, ogni nodo avrà al più un valore associato. Dal punto di vista implementativo si scorrono tutti gli elementi del dizionario, ciascuno dei quali ricordiamo corrisponde ad un tripId, e, per ognuno, si crea un dizionario che ha come chiave l'Id del nodo e come valori il timestamp, il valore di PM10 e la distanza dal nodo della mappa a cui la rilevazione è associata. In questo modo, sempre per un determinato TripId, si potranno ottenere i valori mediani fra tutti i dati associati a ciascun nodo. Il risultato è, ancora una volta, un dizionario che ha come chiave il TripId ma come valore un insieme di array ciascuno dei quali possiede l'Id del nodo associato ad una rilevazione, la data ricavata come valore mediano delle date, la PM10 ricavata anch'essa come valore mediano tra le PM10, la distanza dal nodo della mappa associato ricavato sempre come valore mediano. Un esempio è il seguente:

*{'TRIP_1682698931799': [[316, 1682699074147, 4.0, 0.03], [315, 1682699031964, 6.0, 0.04], [311, 1682698981326, 6, 0.04]], 'TRIP_1682699679064': [[311, 1682699900863, 7.5, 0.04], . . . ] . . . }*

L'operazione successiva consiste nell'estensione dei valori precedenti a tutti i nodi della mappa. Questa, ovviamente, è una semplificazione necessaria in quanto si ha un solo dispositivo di acquisizione dei dati; quindi si hanno pochi dati a disposizione. Se questa fase dovesse essere omessa non si potrebbe simulare, per un determinato giorno, il suggerimento del percorso migliore all'utente. Per quanto riguarda l'implementazione, si parte dall'ultimo dizionario citato e si leggono tutti i valori. Senza entrare troppo nel dettaglio del codice, è sufficiente dire che viene creato un nuovo dizionario popolato percorrendo tutto il grafo e, per ciascun nodo del grafo, viene associato il valore del nodo, vicino cui si è passati con la bici, più vicino. Il nuovo dizionario avrà come chiave l'id di ciascun nodo che forma il grafo e come valore un array composto, a sua volta, da array contenenti la data di acquisizione, i valori di PM10, 0(valore esteso)/1(nodo con dati originali), la distanza dal nodo con dati originali, l'id del nodo con dati originali, l'id del nodo corrente. Semplificando, ciascun nodo ha la lista di tutti i valori ad esso associati e ciascun elemento è relativo ad un trip:

*{0: [[1682698981326, 6, 0, 3.74, 311, 0], [1682699900863, 7.5, 0, 3.74, 311, 0], [1682699940280, 4, 0, 3.72, 310, 0], [1682701624361, 7, 0, 0.67, 24, 0], [1683183556007, 13.0, 0, 0.67, 24, 0], [1683219952344, 11.5, 0, 0.67, 24, 0]], 1: [[1682698981326, 6, 0, 3.72, 311, 1], [1682699900863, 7.5, 0, 3.72, 311, 1], [1682699940280, 4, 0, 3.71, 310, 1], [1682701624361, 7, 0, 0.6, 24, 1], [1683183556007, 13.0, 0, 0.6, 24, 1], [1683219952344, 11.5, 0, 0.6, 24, 1]], . . . }*

A questo punto si vogliono associare a ciascun nodo del grafo i relativi valori. Questo viene fatto confrontando ciascun id dei nodi del grafo con le chiavi dell'ultimo dizionario trattato. Se il match va a buon fine viene creato per quel nodo l'attributo "arduinoData" in cui vengono salvati i relativi valori. Il salvataggio sui nodi del grafo nasce dalla volontà di avere sulla mappa tutti i dati utili. L'ultima operazione di questo capitolo consiste nella costruzione di un file csv per ogni nodo. L'obiettivo è quello di avere, per ciascun nodo, un dato giornaliero, medio, di PM10 dal 1° aprile al giorno antecedente a quello corrente. Oltre al dato medio giornaliero si vogliono ottenere i dati medi di particolato di sei fasce orarie, quindi con ampiezza di quattro ore ciascuna. L'implementazione è contenuta interamente all'interno di un ciclo for che permette di scorrere tutti i nodi del grafo. Questo perché si recuperano i dati acquisiti da Arduino tramite l'attributo "arduinoData" salvato precedentemente per ciascun nodo. A questo punto si crea un numero di file csv pari al numero di nodi del grafo, secondo una nomenclatura del tipo "NodesArduinoData/Node[nodeId].csv". Si settano, quindi, le date di partenza, cioè l'1-4-2023, e di arrivo, ovvero il giorno precedente a quello corrente.

Sfruttando la funzione *timedelta* si incrementa, di volta in volta, il giorno e si aggiunge ad una lista. Quest'ultima alla fine conterrà tutte le date appartenenti all'intervallo precedentemente indicato. Per ciascun giorno verrà confrontata la data che si sta prendendo in esame con tutte le date all'interno della struttura dati nel quale erano stati salvati i dati del nodo in

questione. Ciò permetterà di popolare un array che conterrà tutti i valori relativi ad un certo nodo in un giorno preciso. Stesso ragionamento vale per gli altri sei array rappresentanti le diverse fasce orarie. In quest'ultimo caso il check riguarderà l'appartenenza del dato ad una fascia oraria, piuttosto che ad un'altra, di un determinato giorno. Al termine si calcola la media di ciascun array per avere un unico valore da associare a quel giorno per quel nodo, o a quella fascia oraria per quel nodo. Tale processo non risulta immediato nella comprensione ma l'esempio riportanto nella Tabella 4.1 chiarisce quel è l'obiettivo finale ed il valore apportato da questa operazione.

| **Date** | **Pm10Mean** | **Pm10Mean_0_4** | **Pm10Mean_4_8** |
|---|---|---|---|
| 2023-05-04 | 12.25 | | 13.0 |
| **Pm10Mean_8_12** | **Pm10Mean_12_16** | **Pm10Mean_16_20** | **Pm10Mean_20_24** |
| | | 11.5 | |

**Tabella 4.1:** Esempio di dati relativi al dataset Arduino per il nodo 0

La tabella precedente mostra un esempio di valori del dataset ottenuto per il nodo 0, come si può osservare, il primo campo rappresenta il giorno di interesse (il dataset parte dal 1° aprile 2023 ma nei primi giorni la maggior parte dei dati sono assenti). I campi successivi rappresentano il valore di PM10 per ciascun giorno ed i valori di PM10 registrati nelle fasce orarie 0-4, 4-8, 8-12, 12-16, 16-20, 20-24. Purtroppo, la maggior parte delle righe ha valori vuoti a causa dell'impossibilità di girare continuamente con la bici e a causa della presenza di un solo dispositivo.

In conclusione, si può dire che ciascun nodo della mappa possiede al suo interno sia i dati relativi alle stazioni fisse che quelli relativi al nodo stesso ricavati da Arduino. Questi ultimi sono stati preziosi nella costruzione di un dataset per ciascun nodo che permette di avere a disposizione una evoluzione temporale della PM10.

# CAPITOLO 5

# Servizi per gli utenti

*In questo capitolo verranno presentati quattro servizi messi a disposizione per gli utenti. Tali servizi utilizzeranno quanto visto fino ad ora e, in particolare, sfrutteranno i dati delle stazioni fisse, i dati provenienti da Arduino e la mappa precedentemente creata su NetworkX.*

*Il primo servizio permetterà all'utente di specificare anno, mese, giorno e nodo di interesse. Verranno, così, restituiti, se possibile, i valori di PM10 delle stazioni fisse e di Arduino, relativi alla data indicata. Nel caso in cui la data dovesse essere successiva a quella corrente verrà ritornata la previsione di entrambi. Quest'ultima possibilità sarà limitata alle due settimane successive al giorno in cui il servizio viene usato, per evitare di restituire previsioni poco accurate.*

*Il secondo servizio avrà caratteristiche analoghe al precedente, ma verrà chiesto all'utente di indicare anche l'ora di interesse. Questo avrà impatti sul valore di particolato acquisito da Arduino in quanto, come visto nel capitolo precedente, il dataset è stato costruito tenendo conto anche della fascia oraria.*

*Il terzo, servizio permetterà all'utente di indicare il nodo di partenza, il nodo di arrivo, la data e l'ora. Con queste informazioni verrà calcolato e rappresentato in un grafico, sfruttando i dati provenienti da Arduino, il percorso con i valori di inquinamento più bassi. L'operazione potrà essere richiesta anche per le due settimane successive alla data corrente.*

*Il quarto, ed ultimo, servizio chiederà all'utente di inserire data ed ora. A partire da queste informazioni, verrà costruita una heatmap in cui vengono evidenziate le aree con valori di PM10 alti. Anche per questo servizio è possibile indicare una data futura, rispetttando il limite delle due settimane.*

## 5.1 Premessa

In questo capitolo verranno approfondite le soluzioni adottate per sviluppare i quattro servizi, citati nel preambolo. Verranno, inoltre, mostrati i risultati finali tramite degli esempi.

Prima dello sviluppo dei servizi è stato necessario costruire un menù che permette all'utente di selezionare l'opzione di interesse. In particolare, il menù in questione ha la struttura visibile in Figura 5.1.

```
-------------------------
0. Exit

1. PM10 daily value of a node
2. Value per PM10 timeslot of a node
3. Search less polluted route
4. Plot PM10 heatmap

Choose: 1
------------------------
```

**Figura 5.1:** Menù di scelta servizi

L'implementazione del menù di scelta è molto semplice in quanto consiste in un semplice ciclo while, in cui viene verificato se il numero scelto è diverso da 0. Nel caso affermativo si rimane all'interno del ciclo e, se viene eseguito il match con 1 o 2 o 3 o 4, viene lanciata la funzione, dunque il servizio, corrispondente. Nel caso in cui il numero inserito dall'utente dovesse essere uguale a 0, si esce dal ciclo ed il programma termina.

Prima di procedere è opportuno fare una premessa: a causa della scarsità dei dati ricavati da Arduino, per eseguire i vari test sono stati popolati i dataset di ciascun nodo con dati pseudocasuali. Il valore di ciascun dato è randomico ma all'interno di un intervallo che varia in base al giorno e alla fascia oraria. I campi relativi alle fasce orarie sono stati popolati sulla base dei trend osservati dai dati reali. Pertanto, seppur non costituiscono un caso reale, simulano uno scenario che si avvicina molto alla realtà. Nel momento in cui tale progetto dovesse avere successo, ed essere impiegato su larga scala, non si avrà più la necessità di simulare i dati. Questo perché in una grande città come Torino difficilmente nessuno transiterà vicino ad un nodo per un periodo di 4 ore. Se una volta dovesse decadere tale ipotesi non è un problema in quanto si potrebbe intervenire con un'operazione di interpolazione (per casi eccezionali non verrebbero falsati i risultati in modo significativo).

Ora che è stato chiarito il punto di partenza, si può proseguire con l'analisi dei singoli servizi.

## 5.2 Richiesta dati PM10 per un particolare giorno

Il primo servizio che vediamo permette di ottenere dati sul PM10, passati o futuri, ricavati da stazioni fisse o da Arduino. Tali dati saranno mostrati, se possibile, in seguito alla selezione, da parte dell'utente, della scelta 1 ed in seguito all'indicazione della data e del nodo di interesse.

Prima di mostrare le varie casistiche previste, è necessaria una spiegazione che riguarda l'implementazione e le scelte progettuali adottate.

La prima operazione che viene svolta è la lettura del grafo NetworkX. Questo servirà per eseguire dei controlli sull'esistenza del nodo indicato. Vengono, poi, inizializzate le variabili che verranno associate all'anno, al mese ed al giorno. Con un ciclo while vengono eseguite le seguenti verifiche:

- anno compreso tra 2017 e 2023;
- mese compreso tra 1 e 12;
- giorno compreso tra 1 e 31.

Fino a quando queste condizioni non vengono soddisfatte viene chiesto all'utente di inserire una data, seguendo il formato *YYYY-MM-DD*. Un check sulla validità della data viene eseguito splittando la stringa relativa alla data, in corrispondenza del simbolo -, e creando con tali componenti un oggetto appartenente alla classe `date`. Questo oggetto servirà, anche, per eseguire dei controlli che determineranno alcune casistiche, che vedremo successivamente.

Tramite un altro ciclo while, viene chiesto all'utente di inserire l'identificativo del nodo di interesse. In questo caso le condizioni per poter proseguire sono:

- `nodeId` maggiore o uguale a 0;
- `nodeId` minore o uguale al numero di nodi nel grafo meno uno.

Conseguentemente, si legge il dataset, contenente i dati acquisiti da Arduino, relativo al nodo indicato e si crea il dataframe che possiede come indice il campo *Data*. Stessa cosa viene fatta per il dataset relativo ai dati delle stazioni fisse, settando, però, come indice il campo *DataRilevamento*.

Per gestire i casi che si possono presentare, per i dati provenienti da Arduino, sono stati creati quattro if che tengono conto della data inserita dall'utente:

- la data antecedente il giorno 1° aprile 2023 garantisce l'assenza di dati;
- la data successiva il giorno 1° aprile 2023 e precedente al giorno corrente garantisce la possibilità di leggere il dato dal dataset;
- la data successiva al tredicesimo giorno dalla data corrente non permette di eseguire predizioni accurate;
- la data antecedente al tredicesimo giorno dalla data corrente, e uguale o successiva alla data corrente, permette di eseguire predizioni accurate.

L'ultimo caso è interessante in quanto prevede l'uso di un modello ARIMA (1,0,2) che viene addestrato con i dati a disposizione. La scelta dei parametri del modello è stata eseguita in seguito ad alcuni test.

Per i dati delle stazioni fisse sono stati sviluppati tre if che tengono sempre conto della data inserita dall'utente:

- la data successiva al tredicesimo giorno dalla data corrente non permette di eseguire predizioni accurate;
- la data antecedente al tredicesimo giorno dalla data corrente, e uguale o successiva alla data corrente, permette di eseguire predizioni accurate;
- la data antecedente al giorno corrente garantisce la possibilità di leggere il dato dal dataset.

Anche in questo caso, per la predizione viene sfruttato un modello ARIMA (1,0,2). Inoltre, non è necessaria una data minima, come nel caso dei dati relativi ad Arduino, in quanto il ciclo white iniziale garantisce l'inserimento di una data successiva o uguale al 1° gennaio 2017.

Ora vediamo alcune dimostrazioni del servizio sviluppato.

Nella Figura 5.2 si può osservare come l'utente seleziona il servizio 1 e gli viene chiesto di inserire la data di interesse. In seguito all'inserimento della data *2016-10-10*, viene chiesto all'utente di riprovare in quanto non ci sono dati per date così passate. Con l'inserimento di una data valida, *2023-01-10*, viene chiesto all'utente di inserire il nodo di interesse e viene scelto il nodo *0*. In questo modo si ottiene la scritta *ARDUINO DATA: No data for this date*, in quanto, prima del 1° aprile 2023, non sono stati acquisiti dati, insieme alla scritta *FIXED STATIONS: 14.0*, valore verificato da quanto presente nel dataset delle stazioni fisse.

Un secondo caso, che mostriamo in Figura 5.3, dimostra come, in seguito all'inserimento da parte dell'utente della data *2023-04-10* e del nodo *0*, vengono stampati sia il valore legato ad Arduino che alle stazioni fisse. Si ottengono, rispettivamente, *7* e *9.0*. Nel dataset del nodo 0 ritroviamo *2023-04-10,7,4,5,8,7,10,7*; quindi il secondo valore, relativo al dato giornaliero, ci conferma la correttezza del risultato.

In Figura 5.3 viene mostrato il caso in cui l'utente inserisce una data che supera di tredici giorni la data corrente. In particolare, viene inserita la data *2023-07-10*; a seguito di tale inserimento vengono stampate le scritte *ARDUINO DATA: Prediction is not supported for dates*

```
Choose: 1
------------------------
Write a date following the format YYYY-MM-DD and year starting from 2017: 2016-10-10
Write a date following the format YYYY-MM-DD and year starting from 2017: 2023-01-10
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-01-10 ...

ARDUINO DATA: No data for this date
FIXED STATIONS DATA: 14.0
------------------------
```

**Figura 5.2:** Esempio del servizio 1: data che non rispetta la data minima e assenza di dati in Arduino

```
Choose: 1
------------------------
Write a date following the format YYYY-MM-DD and year starting from 2017: 2023-04-10
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-04-10 ...

ARDUINO DATA: 7
FIXED STATIONS DATA: 9.0
------------------------
```

**Figura 5.3:** Esempio del servizio 1: data in cui sono presenti sia dati delle stazioni fisse che di Arduino

*after* e *FIXED STATIONS DATA: Prediction is not supported for dates after*. È bene ricordare che tutti i test sono stati effettuati in data 12-06-2023.

L'ultimo possibile risultato viene mostrato in Figura 5.5 e riguarda il caso in cui viene inserita una data futura, che però rispetta il vincolo visto precedentemente. L'utente inserisce la data *2023-06-15* e vengono ritornati i valori predetti, *13.91* per Arduino e *20.12* per le stazioni fisse.

Possiamo concludere che il servizio riesce a coprire tutte le possibili casistiche restituendo risultati corretti.

## 5.3 Richiesta dati PM10 per un particolare giorno ed ora

Il secondo servizio che mostriamo è piuttosto simile al primo, in quanto prevede sempre la restituzione dei valori, passati o futuri, delle stazioni fisse e di Arduino. La particolarità risiede nella richiesta all'utente di inserire anche l'ora di interesse. Ciò avrà una influenza sul dato relativo ad Arduino, acquisito o predetto, e si potrà utilizzare per individuare orari critici, dal punto di vista salutare, in particolari nodi.

All'utente non verrà più chiesto il formato *YYYY-MM-DD* per le date ma il formato *YYYY-MM-DD-H*, in cui H indica l'ora. Il resto dell'implementazione è simile al servizio precedente ma, nel momento in cui deve essere restituito un valore, si esegue un check sull'ora. Sulla base dell'esito si prende in considerazione solo il corrispondente campo all'interno del dataframe. Una cosa analoga viene fatta per la fase di predizione, in quanto il modello viene addestrato solo sui dati del dataframe relativi alla fascia oraria indicata.

Vediamo, anche per questo servizio, i test svolti.

In Figura 5.6 viene mostrato il caso in cui l'utente inserisce la data e l'ora *2016-10-10-9*. Viene, quindi, indicato che non ci sono dati provenienti da Arduino per date così lontane nel passato. Il dato delle stazioni fisse, *14.0*, viene ritornato correttamente.

Il secondo caso, visibile in Figura 5.7, mostra come una data compresa tra il 1° aprile 2023 ed il giorno precedente a quello corrente, produce come risultato sia il dato Arduino che quello delle stazioni fisse. Si noti che, rispetto al corrispettivo caso di test del servizio 1, si ha

```
Choose: 1
------------------------
Write a date following the format YYYY-MM-DD and year starting from 2017: 2023-07-10
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-07-10 ...

ARDUINO DATA: Prediction is not supported for dates after 2023-06-25
FIXED STATIONS DATA: Prediction is not supported for dates after 2023-06-25
-------------------------
```

**Figura 5.4:** Esempio del servizio 1: data troppo futura per eseguire predizioni

```
Choose: 1
------------------------
Write a date following the format YYYY-MM-DD and year starting from 2017: 2023-06-15
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-06-15 ...

ARDUINO DATA: (prediction) 13.91
FIXED STATIONS DATA: (prediction) 20.12
-------------------------
```

**Figura 5.5:** Esempio del servizio 1: data in cui è possibile eseguire predizioni

un risultato diverso, dovuto all'orario indicato. Analizzando quanto presente nel dataset del nodo 0, cioè *2023-04-10,7,4,5,8,7,10,7*, si ha che, nel quinto campo, relativo alla fascia oraria 8-12, è presente il valore 8.

Come accade per il servizio 1, in Figura 5.7, viene mostrato il caso in cui viene inserita una data futura, ma che non rispetta il vincolo di massimo 13 giorni rispetto alla data corrente.

In Figura 5.7 viene mostrato il caso in cui la data futura, *2023-01-10-9* , rispetta il vincolo precedentemente esposto. Il risultato che si ottiene sono i valori *16.31*, per Arduino, e *20.12*, per le stazioni fisse. La predizione viene eseguita sulla base dei dati della specifica fascia oraria.

Dalle analisi precedenti si può dedurre che anche il servizio 2 funziona correttamente e gestisce tutti i casi possibili.

## 5.4 Richiesta percorso meno inquinato

Il terzo servizio ha un obiettivo diverso rispetto a quelli visti fino ad ora. Non ci si vuole fermare alla semplice restituzione dei dati, ma si vuole costruire un servizio che possa essere sfruttato quotidianamente dai cittadini di Torino, o di qualsiasi altra grande città. Dalle analisi eseguite nei primi capitoli è emerso che la mobilità tramite bici risulta essere la più green. Inoltre, a Torino, in alcune giornate particolarmente inquinate, si sconsiglia di eseguire attività sportive all'aperto. Per salvaguardare la salute dei cittadini si vuole offrire uno strumento che suggerisce il percorso con livelli di PM10 più bassi. Per poter fare ciò, viene chiesto all'utente il nodo di partenza e di arrivo. La richiesta della data dà la possibilità di restituire sia un percorso basato su dati passati che un percorso basato su valori predetti.

Per quanto riguarda l'implementazione si ha una parte iniziale simile a quanto visto per i primi due servizi. Quindi, vengono eseguiti controlli sulla data ed ora, e vengono chiesti i punti di partenza e di arrivo.

Nel caso in cui tutti i check vengono superati, si può ricadere in due casi: data passata o data futura (con il limite di tredici giorni dalla data corrente). L'unica differenza che distingue queste due casistiche è l'eventuale uso di un modello ARIMA per la predizione. Per i test prendiamo in considerazione solo il caso di data passata.

```
Choose: 2
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2016-10-10-9
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-01-10-9
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-01-10 ed ora 9 ...

ARDUINO DATA: No data for this date
FIXED STATIONS DATA: 14.0
------------------------
```

**Figura 5.6:** Esempio del servizio 2: data che non rispetta la data minima e assenza di dati Arduino

```
Choose: 2
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-04-10-9
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-04-10 ed ora 9 ...

ARDUINO DATA: 8
FIXED STATIONS DATA: 9.0
------------------------
```

**Figura 5.7:** Esempio del servizio 2: data in cui sono presenti sia dati delle stazioni fisse che di Arduino

Viene, innanzitutto, importato il grafo NetworkX e, per ogni nodo, viene letto, dal dataset corrispettivo, il valore di PM10 relativo alla data indicata. I dati estratti vengono salvati in un dizionario che ha come chiave l'identificativo del nodo e come valore il dato stesso. Nel caso di data futura, vengono letti tutti i dati di interesse dal dataframe associato a ciascun nodo e viene predetto il relativo valore. Quest'ultimo verrà sfruttato, anche in questo caso, per la costruzione del dizionario. Vengono poi scansionati tutti gli archi del grafo e, a ciascuno, viene associato, come attributo *Pm10*, il valore calcolato come media dei valori di PM10 dei nodi sottesi. I valori dei nodi sono estratti dal dizionario creato precedentemente. L'ultima operazione prevede l'uso della funzione *shortest_path* che calcola il percorso più breve. Per perseguire l'obiettivo è necessario inserire i seguenti parametri della funzione:

- grafo di riferimento;
- nodo di partenza;
- nodo di arrivo;
- weight, cioè l'attributo degli archi da usare per calcolare il percorso più breve (in questo caso l'attributo Pm10);
- metodo di calcolo del percorso (in questo caso Dijkstra).

Cerchiamo prima di capire in cosa consiste l'algoritmo di Dijkstra. Questo fu inventato nel 1956 dall'informatico olandese Edsger Dijkstra per cercare i cammini minimi in un grafo con o senza ordinamento, ciclico e con pesi non negativi sugli archi. È un algoritmo ad assegnazione di etichetta, cioè i nodi, una volta analizzati ed etichettati, non possono essere più modificati. Questa caratteristica riduce la complessità computazionale dell'algoritmo al numero N dei nodi del grafo.

Ora che è stato salvato il percorso in una variabile, rimane solo da graficare la mappa con il risultato ottenuto. Per una più semplice comprensione della mappa sono stati colorati in arancione i nodi di partenza e di arrivo, in rosso tutti i nodi intermedi appartenenti al percorso e in blu tutti gli altri nodi.

```
Choose: 2
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-07-10-9
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-07-10 ed ora 9 ...

ARDUINO DATA: Prediction is not supported for dates after 2023-06-25
FIXED STATIONS DATA: Prediction is not supported for dates after 2023-06-25
-------------------------
```

**Figura 5.8:** Esempio del servizio 2: data troppo futura per eseguire predizioni

```
Choose: 2
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-06-15-9
Write the node whose value you want to know (from 0 to 344): 0

Calculation of the PM10 value for node 0 on 2023-06-15 ed ora 9 ...

ARDUINO DATA: (prediction) 16.31
FIXED STATIONS DATA: (daily prediction): 20.12
-------------------------
```

**Figura 5.9:** Esempio del servizio 2: data in cui è possibile eseguire predizioni

Anche per questo servizio sono stati eseguiti dei test. In Figura 5.10 viene mostrato il caso in cui viene inserita una data antecedente al 1° aprile 2023 e, come avviene negli altri servizi, viene ritornato il messaggio *ARDUINO DATA: No data for this date*. Una cosa analoga avviene per date troppo future, in cui viene restituito il messaggio *ARDUINO DATA: Prediction is not supported for dates after...*

```
Choose: 3
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-01-10-9
ARDUINO DATA: No data for this date
-------------------------
```

**Figura 5.10:** Esempio di servizio 3: data in cui non sono presenti dati di Arduino

In Figura 5.11 viene mostrato il caso in cui l'utente inserisce una data passata, *2023-04-10-9*, ma che rispetta tutti i vincoli. Vengono chiesti all'utente il nodo di partenza e di arrivo e nel test questo risponde con i valori *10* e *200*. Poi viene visualizzato il messaggio *Calculating best route from node" 10 to node 200 on 2023-4-10 and hour 9 ...* .

Dopo l'elaborazione viene visualizzato il grafo in Figura 5.12 e, come si può notare, il percorso suggerito è credibile in quanto si preferisce passare nella parte destra della mappa. Questa zona è situata in prossimità del fiume Po, dove realisticamente i valori di PM10 sono più bassi.

L'ultimo caso, mostrato in Figura 5.13, vuole simulare l'inserimento di una data futura, che rispetta tutti i vincoli. In questo caso viene indicato come nodo di partenza il *10* e come nodo di arrivo il *340*.

Come si osserva in Figura 5.14, viene, ancora una volta, scelto un percorso che affianca il fiume ed il parco del Valentino. Questa è una simulazione che permette di comprendere i vantaggi che questo servizio comporta per la salute.

Come visto, anche il servizio 3 funziona correttamente ed offre qualcosa di molto utile per i cittadini. Tale servizio riesce, dunque, a perseguire tutti gli obiettivi che erano stati individuati nei capitoli precedenti.

```
Choose: 3
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-04-10-9
Write the index of the starting node (from 0 to 344): 10
Write the index of the arrival node (from 0 to 344): 200

Calculating best route from node 10 to node 200
on 2023-4-10 and hour 9 ...

------------------------
```

**Figura 5.11:** Esempio di servizio 3: data in cui sono presenti dati di Arduino

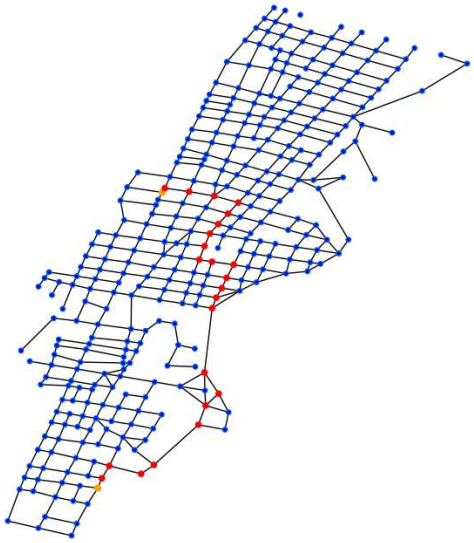

**Figura 5.12:** Esempio di servizio 3: mappa relativa ad una data in cui non sono presenti dati di Arduino

## 5.5 Richiesta heatmap sulla distribuzione di PM10

Il quarto, ed ultimo, servizio ha come obiettivo quello di dare la possibilità all'utente di visualizzare una heatmap, che permette di individuare le aree a maggiore e minore inquinamento. Questo assume un grande valore per le amministrazioni, che devono individuare situazioni critiche in determinate strade o aree. Si possono così sviluppare politiche sia volte all'abbassamento dei valori di PM10, dove necessario, che al miglioramento della gestione del traffico o all'inserimento di aree verdi che possono, in parte, contenere il problema.

La parte iniziale dell'implementazione è simile al servizio 3, ma non viene chiesto all'utente il nodo di partenza e di arrivo. Vengono, però, chiesti la data ed l'ora e, di conseguenza, vengono eseguiti controlli sulla validità di quanto inserito. La verifica permette di inserire date successive il 1° aprile 2023 o superiori alla data corrente di non più di tredici giorni. Allo stesso modo di come accade nel servizio precedente, i dati estratti vengono salvati in un dizionario che ha come chiave l'identificativo del nodo e come valore il dato stesso. È previsto, anche in questo servizio, il caso di data futura. Questo viene gestito secondo le modalità esposte per il servizio 3. Viene, poi, creato un dataset *HeatMapDB* che contiene i campi *Latitude*, *Longitude* e *Pm10*. Ciascuna riga del file contiene i dati relativi ad un nodo della mappa NetworkX. Al termine della scrittura del database, quest'ultimo viene letto e convertito in dataframe *Pandas*. Tramite la funzione *density_mapbox* di plotly (modulo Python che permette una rapida esplorazione dei dati e la generazione di figure) vengono letti i dati del dataframe e viene graficata, usando lo stile *stamen-terrain*, la heatmap.

```
Choose: 3
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-06-15-9
Write the index of the starting node (from 0 to 344): 10
Write the index of the arrival node (from 0 to 344): 340

Calculating best route from node 10 to node 340
on 2023-6-15 and hour 9 ...

-------------------------
```

**Figura 5.13:** Esempio di servizio 3: data futura in cui viene predetto il percorso a partire dai dati di Arduino

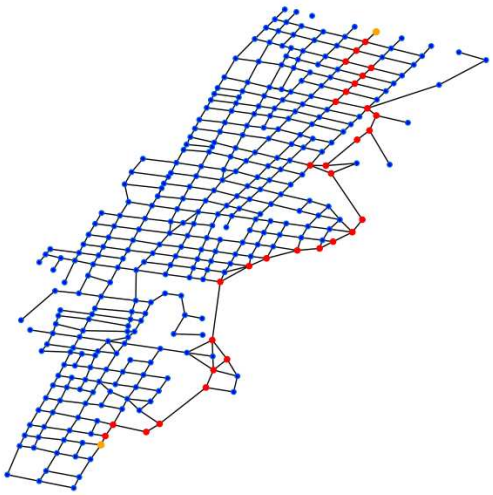

**Figura 5.14:** Esempio di servizio 3: mappa relativa ad una data futura in cui viene predetto il percorso a partire dai dati di Arduino

Spiegato brevemente il funzionamento del servizio, in Figura 5.15 viene mostrato il caso in cui viene inserita una data passata. Non verranno mostrati test relativi ad altre casistiche in quanto sono analoghe a quelle viste negli altri servizi.

```
Choose: 4
------------------------
Write a date following the format YYYY-MM-DD-H and year starting from 2017: 2023-05-10-9

Plot map on 2023-5-10 and hour 9 ...
```

**Figura 5.15:** Esempio di servizio 4: data in cui sono presenti dati Arduino

In Figura 5.16 è osservabile il risultato prodotto; si può notare come le aree con concentrazione di PM10 maggiore sono prevalentemente concentrate in Via Nizza e nell'area del Lingotto. Questo perché, alle 9 di mattina, molte persone si spostano per raggiungere gli uffici presenti in quella zona. L'area a destra della mappa sembra avere una qualità dell'aria migliore; questo è dovuto principalmente alla vicinanza con il parco del Valentino.

Anche il servizio 4 funziona correttamente e, come detto, costituisce uno strumento fondamentale per l'individuazione di aree o strade critiche dal punto di vista salutistico.

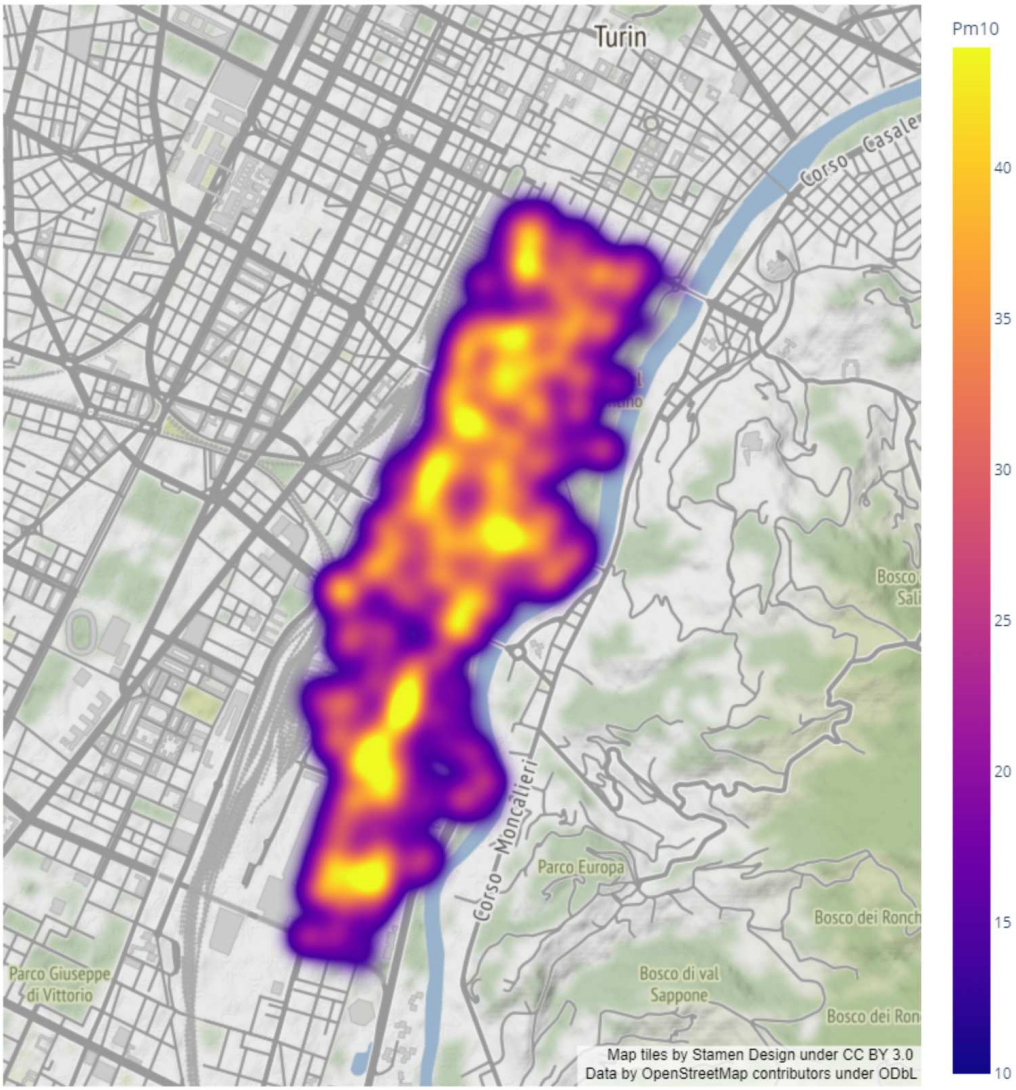


**Figura 5.16:** Esempio di servizio 4: heatmap PM10

# CAPITOLO 6

## Conclusioni e possibile evoluzione del progetto

Il tema centrale dell'intero elaborato, come indicato anche nel titolo, è l'inquinamento nella città. Nel primo capitolo è stato ampiamente spiegato come una qualità dell'aria pessima peggiora sensibilmente la salute delle persone e le espone a rischi, legati all'insorgere di malattie o tumori.

Per contrastare tale fenomeno sono state impiegate varie soluzioni. La prima riguarda l'imposizione, da parte dell'UE, o il suggerimento, da parte dell'OMS, di limiti riguardanti vari agenti inquinanti. Questa soluzione si è rivelata in parte efficace, perché le sanzioni costituiscono sempre un deterrente, ma non risolve alla radice il problema. Un fattore che ha permesso di migliorare la qualità dell'aria, nei Paesi più sviluppati, è stata l'introduzione di filtri antiparticolato e lo sviluppo di nuovi motori, termici ed elettrici. Si parla di motori in quanto, nelle città, una delle principali cause dell'inquinamento è il traffico. In quest'ottica, da decenni, sono state trovate soluzioni alternative che hanno emissioni molto basse, come la metro. Precedentemente è stato precisato che la qualità dell'aria è migliorata nei Paesi più sviluppati; infatti, questo cambiamento non è sempre attuabile in tutti i Paesi. Molto spesso, la causa di questo mancato progresso sta nella necessità di forti investimenti, che non sempre sono possibili.

Il problema dell'inquinamento non impatta in ugual modo tutte le aree del mondo. Per esempio, la Pianura Padana è tra le aree più inquinate al mondo, sia per ragioni morfologiche del territorio che per il grande sviluppo industriale nell'area.

Fino ad ora, sono state spese molte parole per far comprendere l'importanza di questa tematica. Molto spesso, però, una immagine permette di trasmettere la gravità della situazione, in particolari aree, in modo più rapido ed efficace. Quindi, in Figura 6.1, viene mostrata la distribuzione dell'inquinamento in Europa, negli ultimi 18 mesi. È immediatamente visibile come la situazione nel Nord Italia sia critica.

Poi, nel Capitolo 2, è stata eseguita un'analisi sulla correlazione tra il PM10 ed i fattori meteorologici per la città di Torino. I dati sono stati raccolti direttamente dal sito dell'ARPA Piemonte ed è emerso che:

- la temperatura influenza fortemente la quantità di PM10, secondo una correlazione inversamente proporzionale;
- il vento ha un effetto benefico sui livelli di PM10 solo se supera una velocità minima;
- la pioggia ha un influenza piuttosto moderata sul PM10.

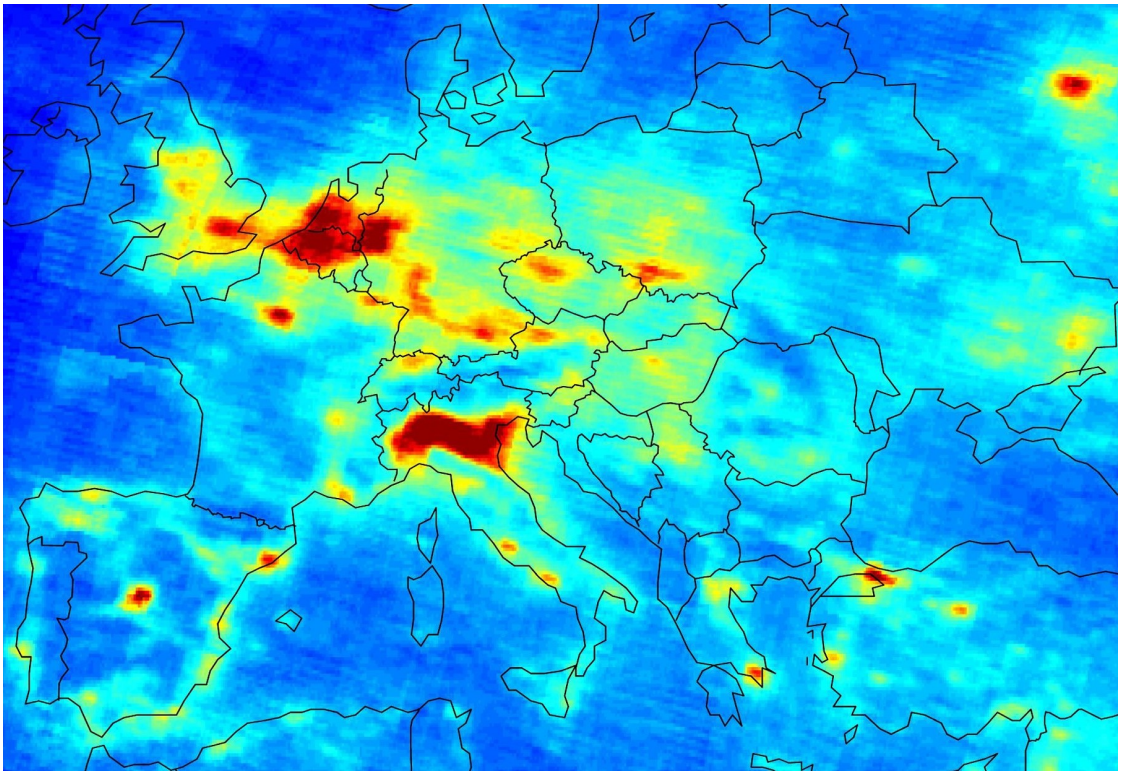

**Figura 6.1:** Mappa ESA sulla distribuzione di inquinamento in Europa

Oltre a fornire informazioni utili, questa analisi potrebbe essere sfruttata per migliorare i modelli previsionali sulla PM10. Parleremo di questo punto successivamente.

Per poter raccogliere dati puntuali, che permetterebbero di capire in che strade si concentrano valori alti di PM10, è stato creato un dispositivo basato su Arduino. Nel Capitolo 3 sono state illustrate tutte le componenti che hanno permesso di realizzarlo, fra cui il sensore GPS ed il sensore PM10. Il tutto è stato confezionato all'interno di una scatola di derivazione esterna, coperta da un pannello in plexiglass. Tale scatola è stata forata per permettere di alimentare Arduino e per avere un flusso di aria ottimale all'interno della stessa. Questo dispositivo, come spiegato nel relativo capitolo, è stato realizzato con l'idea di essere installato su biciclette e monopattini; quindi si è posta particolare attenzione sulle dimensioni del prodotto. Il codice sviluppato ha permesso di rendere il tutto pienamente funzionamente. Sono state adottate soluzioni all'avanguardia, come il salvataggio real time su database, dopo aver superato diverse fasi di elaborazione su NodeRED.

Nel Capitolo 4 è stato spiegato come, per costruire il servizio che permette di suggerire all'utente il percorso migliore dal punto di vista salutare, è stato necessario costruire un grafo NetworkX di un quartiere di Torino. A ciascun nodo sono stati associati sia i valori delle stazioni fisse che dei dati acquisiti da Arduino. Per perseguire quest'ultimo obiettivo, è stato necessario connettersi al database CoutchDB. In seguito a varie operazioni svolte sui dati, sono stati salvati nel grafo quelli relativi a ciascun nodo. A partire da questo, è stato costruito un dataset per ciascun nodo, che permettesse di avere uno storico, ed eventualmente eseguire predizioni.

Infine, nel Capitolo 5 sono stati mostrati i quattro servizi, che ricordiamo essere:

- Servizio 1: permette all'utente di inserire data e nodo di interesse ed ottenere il relativo valore di PM10.

- Servizio 2: permette all'utente di inserire data, ora e nodo di interesse ed ottenere il relativo valore di PM10.
- Servizio 3: permette all'utente di inserire data, ora, nodo di partenza, nodo di arrivo ed ottenere il percorso con livelli di PM10 più bassi.
- Servizio 4: permette all'utente di inserire data ed ora e visualizzare la heatmap sulla distribuzione di PM10.

Tali servizi sono accessibili grazie ad un menù a scelta e prevedono la possibilità di restituire risultati che riguardano sia date passate che future, tramite predizione.

Questo progetto può costituire uno strumento importante per le amministrazioni, in quanto queste possono individuare aree con concentrazioni di PM10 particolarmente alte e, sulla base delle informazioni acquisite, apportare delle modifiche alla viabilità. Anche i cittadini possono sfruttare i servizi messi a disposizione, ma per rendere il prodotto fruibile e commercializzabile sono necessarie delle evoluzioni.

Una prima evoluzione potrebbe permettere il recupero dei dati delle stazioni fisse in modo automatico. Attualmente questo non è possibile in quanto è necessario eseguire richieste specifiche ad ARPA Piemonte. ARPA restituisce così dei file `.csv` contenenti i dati fino alla data della richiesta, o alla data indicata. Ovviamente, sarebbero necessarie richieste particolari volte ad accedere direttamente al loro database.

Una seconda evoluzione riguarda l'eliminazione dell'algoritmo che estende il valore associato ai nodi, in cui si è passati con la bici, ai nodi limitrofi. Questo algoritmo è stato necessario in quanto si ha a disposizione un solo dispositivo, risultando impossibile passare per tutte le strade. Quindi, in ottica di produrre servizi e test funzionanti, si è deciso di adottare provvisoriamente questa soluzione.

La terza evoluzione consiste nell'eliminazione dell'algoritmo che provvede alla generazione dei database, dei vari nodi della mappa, contenenti i dati Arduino. La ragione per il quale è stato inserito l'algoritmo è analoga al punto precedente, ovvero la presenza di pochi dati e l'impossibilità di eseguire test dei servizi. In un caso reale, ovviamente, non si presenterebbe questo problema e, qualora dovesse accadere che un dato non è presente, si potrebbe intervenire con una interpolazione dei dati. L'interpolazione, che interverrebbe su pochi valori nulli, non impatterebbe in modo significativo sulla correttezza dei risultati.

Un'altra evoluzione prevede la costruzione di applicazioni Android e IOS, che permetterebbe di rendere il servizio accessibile a tutti. Per perseguire tale obiettivo sarebbe necessario sviluppare una interfaccia, e quindi un frontend, "user friendly". Questo perché nel tempo gli utenti danno sempre più importanza all'aspetto e alla semplicità d'uso di un'app. Si potrebbe pensare ad una interfaccia in un cui l'utente non deve necessariamente inserire il nodeId, per i servizi 1, 2 e 3, ma può selezionare da una mappa i nodi di interesse.

L'ultima, ma non meno importante, evoluzione potrebbe permettere di estendere la mappa, quindi il grafo NetworkX, all'interna città di Torino, e non solo. Si potrebbe, infatti, pensare di implementare una rete neurale che, tramite algoritmi di computer vision e deep learning, riesca a riconoscere in modo automatico i nodi della mappa. In input riceverebbe la mappa di una città ed in output restituirebbe la lista dei nodi con le relative coordinate geografiche.

È doveroso precisare che la bontà del servizio dipende fortemente dalla quantità di dati che vengono raccolti. Quindi, sarebbero necessarie politiche che incentivino le società di sharing, di bici e monopattini, ad implementare il dispositivo nei loro mezzi. Questo permetterebbe anche di superare il problema della connessione Wi-Fi con il telefono dell'utente, in quanto ci si può appoggiare alla connessione del mezzo stesso.

# Bibliografia


WHALEY, P., NIEUWENHUIJSEN, M. e BURNS, J. (2022), *Update of the WHO Global Air Quality Guidelines: Systematic Reviews*, Environment International.

ASSEMBLY W. H e SCHROEDER, C. H. (2015), *Health and the environment: addressing the health impact of air pollution*, World Health Organization's Institutional Repository for Information Sharing.

CITTÀ DI TORINO (2013), *Piano della mobilità ciclabile*.

MANISALIDIS, I., STAVROPOULOU, E., STAVROPOULOU, A. e BEZIRTZOGLOU, E. (2020), *Environmental and Health Impacts of Air Pollution*, Frontiers.

SICARD, P., AGATHOKLEOUS, E., DE MARCO, A., PAOLETTI, E. e CALATAYUD, V. (2021), *Urban population exposure to air pollution in Europe over the last decades*, SpringerOpen.

LIU, Y., ZHOU, Y.e LU, P. (2020), *Exploring the relationship between air pollution and meteorological conditions in China under environmental governance*, National Library of Medicine.

SWART, P. (2008), *Exploring Network Structure, Dynamics, and Function Using NetworkX*, ResearchGate.

ARTLEY, B. (2022), *A deep-dive on the gold standard of time series forecasting*, Medium.

BEATLEY, T. (2012), *Green Cities of Europe*, Island Press Washington, DC.

FERENCZ, K. e DOMOKOS, J. (2020), *Using Node-RED platform in an industrial environment*, ResearchGate.

BAKOLIS, I., HAMMOUD, R., STEWART, R., BEEVERS, S., DAJNAK, D., MACCRIMMON, S., BROADBENT, M., PRITCHARD, M., SHIODE, N., FECHT, D., GULLIVER, J., HOTOPF, M., HATCH, S. L.e MUDWAY, I. S. (2020), *Mental health consequences of urban air pollution: prospective population-based longitudinal survey*, Springer Link.

LITMAN, T. (2021), *Smart Transportation Emission Reduction Strategies*, Victoria Transport Policy Institute.

## Siti web consultati


- Aria Ambiente Piemonte, dati qualità dell'aria – `aria.ambiente.piemonte.it`
- Arpa Piemonte, dati meteorologici – `www.arpa.piemonte.it`
- Adafruit – `www.adafruit.com`
- Arduino Store – `store.arduino.cc`
- Ansa Motori – `www.ansa.it/canale_motori`
- Ilsole24ore – `www.ilsole24ore.com`
- Geo Net srl – `apps.who.int`
- Istituto Superiore di Sanità – `www.epicentro.iss.it`
- Climate by Selectra – `www.climate.selectra.com`
- IQAir – `www.iqair.com`
- European Environment Agency – `www.eea.europa.eu`
- Our World in Data – `www.ourworldindata.org`
- Infrato – `www.infrato.it`
- Viviamo Sostenibile – `www.viviamosostenibile.it`
- Sixt – `www.sixt.it`
- Comune di Torino – `www.comune.torino.it/`
- Corte dei Conti Europea – `www.eca.europa.eu`
- Gruppo Sanpellegrino – `www.sanpellegrino-corporate.it`
- Fondazione Veronesi – `www.fondazioneveronesi.it`
- Primo Piano – `www.primo-piano.info`
- European Space Agency – `www.esa.int`

# Ringraziamenti

In questa sezione della tesi vorrei spendere qualche parola per tutti coloro che hanno contribuito alla realizzazione della stessa.

Il primo ringraziamento va al Prof. Ursino, che mi ha accompagnato nel lavoro di tesi, seguendomi con estrema disponibilità e comprensione.

Un ringraziamento speciale va ai miei genitori, che mi hanno dato la possibilità di intraprendere e portare a termine questo percorso. Sono stati al mio fianco e mi hanno sostenuto in tutti i momenti di difficoltà incontrati in questi ultimi mesi.

Ringrazio Blue Reply ed i manager Andrea Venditti e Alessandro Chiereghin, che si sono interessati al progetto ed hanno contribuito finanziariamente alla realizzazione dello stesso.

Un grazie infinito va a Matteo Nisi e Umberto Sgueglia, che si sono spesi attivamente nello sviluppo della tesi. Senza di loro la realizzazione della stessa ed il superamento di alcune problematiche non sarebbero stati possibile.

Ringrazio tutti i miei amici, con cui ho condiviso tutte le esperieze più importanti della vita e che, anche durante il periodo di realizzazione della tesi, mi hanno regalato momenti di spensieratezza.